Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nejc Novak

Posebnosti razvoja aplikacij za Windows Phone 8

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentorica: doc. dr. Mira Trebar

Ljubljana 2014

Rezultati diplomskega dela so intelektualna lastnina avtorja. Za objavljanje ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje avtorja, Fakultete za računalništvo in informatiko ter mentorja

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Mobilne ali pametne naprave omogočajo uporabnikom hitro in učinkovito pridobivanje in izmenjavo informacij. Pri razvoju aplikacij je potrebno upoštevati različne pristope in definirati postopke, ki bodo razvijalca vodili do kakovostne in uspešne rešitve. Na trgu se nahajajo številni ponudniki naprav z različnimi operacijskimi sistemi, kjer so najbolj razširjeni iOS, Android in Windows Phone 8.

Kandidat naj v diplomskem delu analizira ustrezne tehnologije, principe in postopke, ki jih je potrebno upoštevati pri celotnem procesu izdelave priljubljene in tržno uspešne mobilne aplikacije. Za Windows Phone 8 naj predstavi posebnosti razvoja na primeru interaktivne aplikacije za izmenjavo informacij, ki vključuje vsa potrebna ogrodja, orodja in programske rešitve.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Nejc Novak, z vpisno številko **63100014**, sem avtor diplomskega dela z naslovom:

Posebnosti razvoja aplikacij za Windows Phone 8

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Mire Trebar,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 22. avgust 2014 Podpis avtorja:

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
ARGB	Alpha Red Green Blue	alfa rdeče zeleno modro
ASO	Application Store Optimization	optimizacija za trg aplikacij
BLOB	Binary Large Object	večji binarni objekt
CSS	Cascading Style Sheets	kaskadne stilske podloge
HTML	Hyper Text Markup Language	označevalni jezik za nadbesedilo
KPI	Key Performance Indicator	ključni pokazatelji uspešnosti
LINQ	Language-Integrated Query	jezikovno-integrirano povpraševanje
SDK	Software Development Kit	programska oprema za razvijanje
SQL	Structured Query Language	strukturirani povpraševalni jezik
WP8	Windows Phone 8	Windows Phone 8
WP8.1	Windows Phone 8.1	Windows Phone 8.1
XAML	Ext. Application Markup Lang.	razširljiv označevalni jezik
XML	Extensible Markup Language	razširljiv označevalni jezik

Povzetek

Na svetu je več kot 1.5 miljarde uporabnikov pametnih telefonov. Uporaba mobilnih aplikacij stalno narašča, vendar jih večina ne uspe. Na trgu jih obstaja že več milijonov, standardi uporabnikov pa so vedno višji. Cilj diplomskega dela je predstavitev izdelave uporabnih in uspešnih aplikacij za Windows Phone 8 (WP8). Zbrani so bili podatki o razvoju aplikacij za ta sistem in posebnostih, ki ga razlikujejo od ostalih. V smiselno celoto so bili združeni s procesi, ki so potrebni za razvoj kvalitetnih produktov, a so pri razvoju aplikacij pogosto zanemarjeni. Teoretična razlaga je razširjena s primeri iz aplikacije "Shop or Drop". Namenjena je ljudem, ki nakupujejo oblačila ter potrebujejo mnenja ostalih. Na enostaven način se slikajo v oblačilu in sliko pošljejo prijateljem. S pomočjo pridobljenih mnenj se lažje odločijo ali bodo oblačilo kupili. V aplikaciji so bile upoštevane posebnosti sistema WP8, razvita pa je bila v skladu s postopki, ki so predstavljeni v diplomskem delu.

Ključne besede: pametni telefon, mobilna aplikacija, Windows Phone 8.

Abstract

There are more than 1.5 billion smartphone users and the use of mobile applications (apps) is steadily increasing. Most never succeed because of users' high expectations and market saturation. The aim of this thesis is to present the process of creating useful and successful apps for Windows Phone 8 (WP8). The data presented was collected with regard to the specifics that differentiate WP8 from other operating systems. App developers often disregard processes that should be present in the development of any quality product. They were merged with the data about app development. The theory is expanded with examples from the app "Shop or Drop". It was developed for people that need help deciding while clothes shopping. They can simply take a picture of themselves in the chosen clothes and send it to their friends. Their opinions are collected in the app and that helps the user to decide. It was developed in accordance with the processes and specifics of WP8 explained in the thesis.

Keywords: smartphone, mobile application, Windows Phone 8.

Poglavje 1

Uvod

Mobilni telefoni so temeljito spremenili način komunikacije med ljudmi. Omogočajo opravljanje klicev ne glede na to kje smo. Pametni mobilni telefoni pa omogočajo mnogo več kot samo opravljanje klicev. Uporabniki imajo na voljo dostop do interneta, navigacijo, poslušanje glasbe, igranje igric in najpomembneje - uporaba aplikacij. Ocenjuje se, da je na svetu 4.4 miljarde uporabnikov mobilnih telefonov. Od tega je skoraj 40% pametnih telefonov. Revolucija pametnih telefonov in aplikacij se je pričela z operacijskima sistemoma Android in iOs, nadaljuje pa se z Windows Phone. Windows Phone Store postaja vedno bolj perspektiven trg za trženje aplikacij, saj še ni tako zasičen kot AppStore za iOS in Google Play za Android. Število razvijalcev in aplikacij za Windows Phone se v zadnjih letih vztrajno veča. Trend rasti števila uporabnikov in uporaba na mobilnih napravah proizvajalcev kot so HTC, Samsung, Nokia in Huawei pa nakazujeta, da postaja Windows Phone resen tekmec na trgu mobilnih operacijskih sistemov.

Cilj diplomskega dela je predstavitev celostnega pristopa pri izdelavi aplikacij za mobilni operacijski sistem Windows Phone 8 (WP8). Začetniki, ki želijo razvijati za WP8, težko najdejo ustrezen vir, ki bi opisoval celoten proces izdelave aplikacije. Spletne strani podjetja Microsoft so zanesljiv vir s kvalitetnimi materiali. Vendar pa so količine podatkov z opisi zelo obsežne, osredotočajo pa se samo na razvoj[1]. Tudi ostali viri ne ponujajo podatkov, ki bi poleg razvoja opisovali še

[1]Beseda razvoj je v tem diplomskem delu uporabljena za oblikovanje in programiranje aplikacije in storitev, ki jih potrebuje. To je del procesa izdelave aplikacije, ki pride na vrsto po zasnovi in validaciji ideje ter analizi trga.

ostale pomembne dele izdelave aplikacij. Pred začetkom razvoja so potrebni analiza problema, zasnova ter validacija ideje za reševanje tega problema, upoštevanje potreb uporabnikov, kvaliteten poslovni načrt in prototip. Da razvita aplikacija doseže svoj poln potencial pa je potrebno po razvoju izvesti testiranje, vzpostaviti analitiko, vzdrževati komunikacijo z uporabniki in aplikacijo z različnimi metodami tržiti. Brez teh postopkov aplikacija ne bo uspešna in uporabna, težko pa bo dosegla čim večji del ciljne populacije. V tem diplomskem delu je zato predstavljen celoten proces izdelave aplikacije, vključno s prej naštetimi postopki. Podatki so zbrani iz različnih virov, združeni in predstavljeni na zgoščen način. Podprti pa so tudi s primerom praktične aplikacije "Shop or Drop". Osrednji del opisuje razvoj, ker je ta del večinoma najbolj časovno zahteven. Povdarek pri razvoju pa je na posebnostih WP8, ki so značilne samo za njega. Razvijalci operacijskega sistema WP8 so lahko temeljito preučili prednosti in slabosti ostalih, starejših operacijskih sistemov. Prednosti so uporabili, slabosti pa popravili. Z upoštevanjem posebnosti WP8 ter celostnim pristopom je mogoče ustvariti popularne, uporabne in uspešne aplikacije.

Poglavje 2

Uporabljene tehnologije in orodja

Pri razvoju aplikacij za mobilni operacijski sistem WP8 je potrebno poznavanje različnih tehnologij, programskih jezikov in orodij. Potrebna pa je tudi določena strojna in programska oprema. V tem poglavju so opisane vse tehnologije in orodja, ki so ključna pri razvijanju aplikacij.

2.1 Windows Phone 8

2.1.1 Osnovni podatki

WP8 je različica mobilnega operacijskega sistema Windows Phone. Podjetje Microsoft je WP8 izdalo 29. oktobra 2012. Je tretja in najnovejša različica družine Windows Phone operacijskih sistemov. Obstaja tudi WP8.1, vendar je trenutno na voljo samo za razvijalce [4]. Microsoft jo je predstavil 2. aprila 2014 na konferenci Microsoft Build, za vse uporabnike naprav z operacijskimi sistemi Windows Phone pa naj bi bila dostopna nekaj mesecev po predstavitvi.

Windows Phone je tretji najbolj razširjen mobilni operacijski sistem. Ocenjuje se, da je njegov tržni delež okoli 3% [5]. Še vedno močno zaostaja za mobilnimi operacijskimi sistemi Android in iOS, vendar se hitro razvija in postaja vse bolj priljubljen. Na Windows Phone Store se nahaja že več kot dvesto tisoč aplikacij, med njimi že tudi večina večjih imen v industriji mobilnih aplikacij in iger [6].

2.1.2 Pogoji za razvijanje aplikacij

Za razvoj WP8 aplikacij je potreben računalnik, na katerem je nameščen operacijski sistem Windows 8 in programska oprema Visual Studio. Različica Visual Studio Professional ima Windows Phone SDK (Software Development Kit) 8.0 že integriran, drugače pa ga je potrebno posebej namestiti.

Emulator omogoča kreiranje navideznih mobilnih naprav, na katerih se lahko testirajo razvite aplikacije. Za uporabo WP8 emulatorja je potrebno vsaj 4GB pomnilnika in ena izmed boljših različic operacijskega sistema Windows 8 (vsaj Windows 8 Pro) [7]. Aplikacije je mogoče testirati tudi na mobilnih napravah, ki imajo nameščen WP8. Ta možnost je na voljo samo razvijalcem, saj mora biti mobilna naprava prijavljena kot testna naprava. Vsak razvijalec ima lahko hkrati registrirane tri testne naprave. Zaključene aplikacije lahko izdajo na Windows Phone Store, kjer so na voljo vsem uporabnikom WP8.

2.1.3 Uporabniški vmesnik

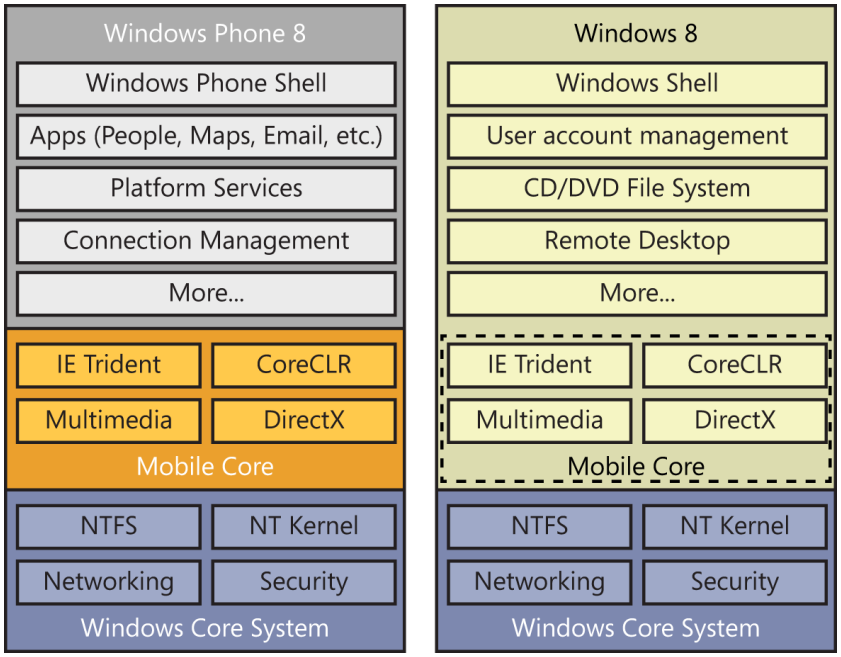
Uporabniški vmesnik v WP8 je osnovan na oblikovalskem jeziku Metro. Po videzu se močno razlikuje od uporabniških vmesnikov sistemov Android in iOS. Velik poudarek je na dostopnosti informacij. Navadne ikone nadomestijo žive ploščice (ang. Live Tiles), ki niso samo statične bližnjice do aplikacij ali določenih delov aplikacij. Lahko tudi dinamično prikazujejo podatke v realnem času. Primer je aplikacija Koledar (ang. Calendar). Na živi ploščici so vidni pomembni podatki za prihodnje dni. To lahko uporabniku prihrani čas, ki je potreben za zagon aplikacije in preverjanje teh podatkov v aplikaciji (slika 2.1a). Uporabnik lahko spremeni privzeto barvo ploščic in ploščice dodaja, odstranjuje, premika ali spreminja njihovo velikost. Velikost ploščic vpliva na količino informacij prikazanih na posamezni ploščici. Na voljo so tri velikosti, s tem da najmanjša ne prikazuje nobenih podatkov in služi samo zagonu aplikacije. Podobno kot na živih ploščicah so tudi na zaslonu za odklep telefona lahko prikazani pomembni podatke, ki jih uporabnik vidi še preden telefon odklene (slika 2.1b).

Poleg živih ploščic se WP8 razlikuje od ostalih mobilnih operacijskih sistemov tudi po razdelkih aplikacij (ang. Hubs). Ti so posebnost, saj združujejo več aplikacij s podobno temo in vsebino. Iz razdelkov lahko upravljamo z več aplikacijami

(a) Začetni zaslon (b) Zaslon za odklep

Slika 2.1: Uporabniški vmesnik WP8

naenkrat. Primer je razdelek s slikami (ang. Pictures Hub). V tem razdelku so slike zajete s pomočjo kamere na mobilni napravi, slike s socialnih omrežij in storitev, ki omogočajo shranjevanje slik. Na napravah so na voljo še ostali razdelki. Razdelek s kontakti (ang. People Hub), povezuje kontakte iz imenika, socialnih omrežij in elektronskih računov. Razdelek z glasbo in videom (ang. Music + Videos Hub) združuje aplikacije, ki služijo prikazovanju glasbe in videov. Razdelek z igrami (ang. Games Hub) vsebuje vse igre, ki so nameščene na mobilni napravi.

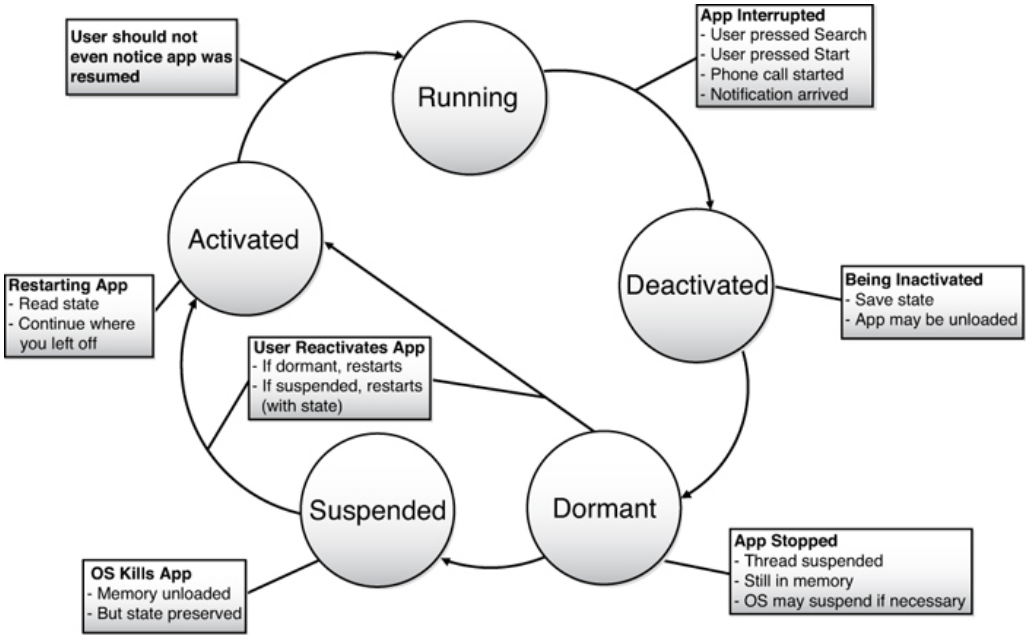


Slika 2.2: Primerjava arhitekture Windows Phone 8 in Windows 8

2.1.4 Arhitektura

WP8 je prvi mobilni operacijski sistem v družini Windows Phone, katerega arhitektura temelji na Windows NT tehnologiji. Jedro Windows CE, ki ga je uporabljal Windows Phone 7, je nadomestilo jedro NT. WP8 uporablja enak datotečni sistem, jedro NT, varnostne in omrežne elemente kot Windows 8 (slika 2.2). Zaradi tega se lahko aplikacije za Windows 8 dosti lažje priredijo za WP8 in obratno. Jedro NT nudi mnogo nadgradenj od prejšnega jedra CE. Med drugim omogoča podporo za večjedrne procesorje (do 64 jeder), višje resolucije (do 1080p oziroma 1920x1080) in spominske kartice MicroSD.

Poleg jedra NT je del arhitekture tudi mobilno jedro. Vsebuje sklop funkcij sistema Windows, ki niso del jedra NT, vendar so pomembne za mobilne naprave. Vse te funkcije so prisotne tudi v sistemu Windows 8, vendar so le del večjega sklopa funkcionalnosti (to je na sliki 2.2 ponazorjeno s prekinjeno črto). Aplikacije so nameščene na delu operacijskega sistema, ki je arhitekturno gledano nad obema jedroma. Vsebuje pa tudi storitve za upravljanje z operacijskim sistemom in aplikacijami ter ostala orodja.



Slika 2.3: Življenjski cikel aplikacije [1]

2.1.5 Življenjski cikel aplikacije

Zaradi omejenih zmožnosti mobilnih naprav in boljše uporabniške izkušnje na WP8 je naenkrat aktivna samo ena aplikacija. Ostale so medtem v mirovanju. Njihovo delovanje pa se nadaljuje, če pridejo v ospredje. Ko se mora aplikacija, ki ni v ospredju ustaviti, se podatki lahko shranijo in ob ponovnem zagonu naložijo v aplikacijo. Uporabnik ima tako vtis, da se vse aplikacije izvajajo ves čas [1].

Življenjski cikel aplikacije vsebuje 5 stanj (slika 2.3). Ko je aplikacija zagnana je v stanju aktivacije (ang. Activated), kjer se preveri če obstajajo shranjeni podatki. Če obstajajo, se naložijo v aplikacijo. Dokler je aplikacija v ospredju je v stanju izvajanja (ang. Running). Če pride v ospredje druga aplikacija ali pa je trenutna aplikacija na drug način prekinjena, gre v stanje deaktivacije (ang. Deactivated). V tem stanju se shranijo morebitni podatki, ki bodo potrebni ob ponovni aktivaciji. Iz stanja deaktivacije aplikacija kasneje preide tudi v stanje mirovanja (ang. Dormant). V tem stanju je še vedno v spominu, operacijski sistem pa jo lahko ob pomanjkanju spomina zaustavi in odstrani iz spomina (ang. Suspended). Če uporabnik ponovno zažene aplikacijo ali se vanjo vrne, se iz stanja mirovanja povrne v stanje aktivacije.

2.2 Programski jezik C#

C# je programski jezik, ki ga je razvilo podjetje Microsoft. Na voljo je od leta 2000. Cilj razvijalcev je bil razviti moderen, preprost, splošno uporaben, objektno orientiran jezik [8]. Vsebuje mnogo značilnosti jezikov C, C++ in Java. Kot večina novejših programskih jezikov je C# objektno orientiran, po sintaksi pa najbolj podoben Javi. Je najbolj pogosto uporabljen jezik pri razvoju za sisteme Windows in Windows Phone. Glede na TIOBE indeks je na šestem mestu na lestvici najbolj popularnih programskih jezikov [9].

2.3 XAML

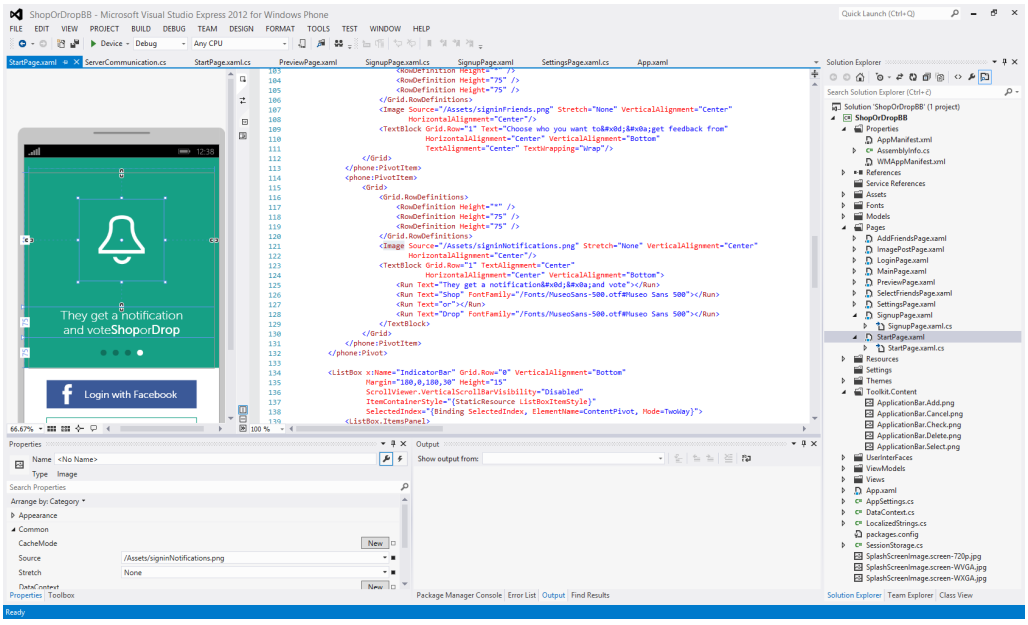
XAML (Extensible Application Markup Language) je deklarativen označevalen jezik osnovan na jeziku XML (Extensible Markup Language). Razvilo ga je podjetje Microsoft, leta 2008, za uporabo pri svojih .NET ogrodjih in ostalih tehnologijah. Uporablja se za opisovanje uporabniškega vmesnika, ki je ločen od ostalih delov aplikacije oz. programa. Z uporabo C# se lahko implementira vse, kar je mogoče narediti v jeziku XAML. Primer gumba ustvarjenega s pomočjo XAML:

```
<Button Width="100"
        Content="I am a Button"
        Background="LightGray" />
```

Isti gumb z uporabo programskega jezika C#:

```
Button b1 = new Button();
b1.Width = 100;
b1.Content = "I am a Button";
b1.Background = new SolidBrushColor(Colors.LightGray);
```

Vseeno je XAML za opisovanje uporabniškega vmesnika boljša izbira. Z uporabo XAML je uporabniški vmesnik na svojem, ločenem nivoju. Hkrati pa XAML, zaradi svoje strukturiranosti in osnovanosti na XML jeziku, zmanjša kompleksnost ter izboljša berljivost in preglednost kode [10].



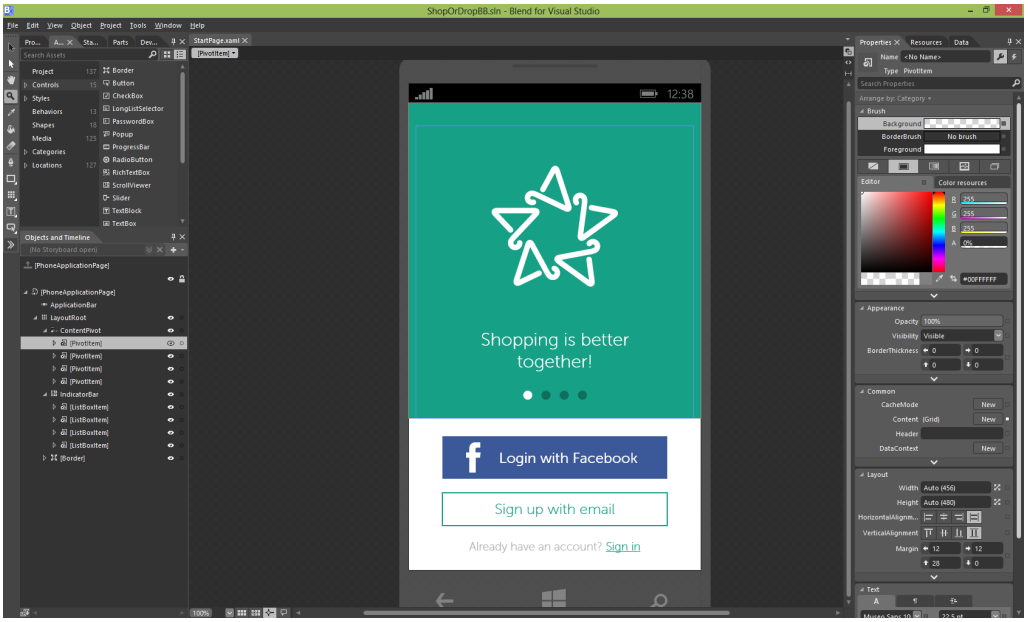
Slika 2.4: Razvojno okolje Visual Studio

2.4 Visual Studio

Visual Studio je razvojno okolje, ki ga je razvilo podjetje Microsoft. Prva različica je bila razvita že leta 1995, trenutno najnovejša verzija pa je Visual Studio 2013. Namenjen je predvsem razvoju programov in aplikacij za Windows in Windows Phone ter spletne strani, aplikacije in storitve. Podprta je uporaba jezikov C, C++, Visual Basic, C#, F#, Python, Ruby in tudi JavaScript, CSS, XML in HTML. Za razvoj WP8 aplikacij obstaja Visual Studio Express for Windows Phone. Ta različica ne vsebuje vseh funkcionalnosti Visual Studia, vendar ima vsa potrebna razvijalska orodja in Windows Phone SDK [11].

Visual Studio ponuja funkcije in orodja, ki olajšajo delo razvijalcu. Med drugim omogoča pregled možnih ukazov, samodejno vstavljanje, dokončevanje in refaktoriranje kode. Ima vgrajen razhroščevalnik in emulator. V razhroščevalniku se lahko določijo točke ustavitve. Te omogočajo preverjanje vrednosti spremenljivk in objektov med delovanjem programa ali aplikacije, kar olajša iskanje napak v kodi. Emulator je namenjen testiranju videza in delovanja aplikacij. Olajša testiranje delovanja pri različnih resolucijah in spominskih zmogljivostih. Testiranje na emulatorju je priporočljivo, vendar je potrebno aplikacije vedno testirati tudi

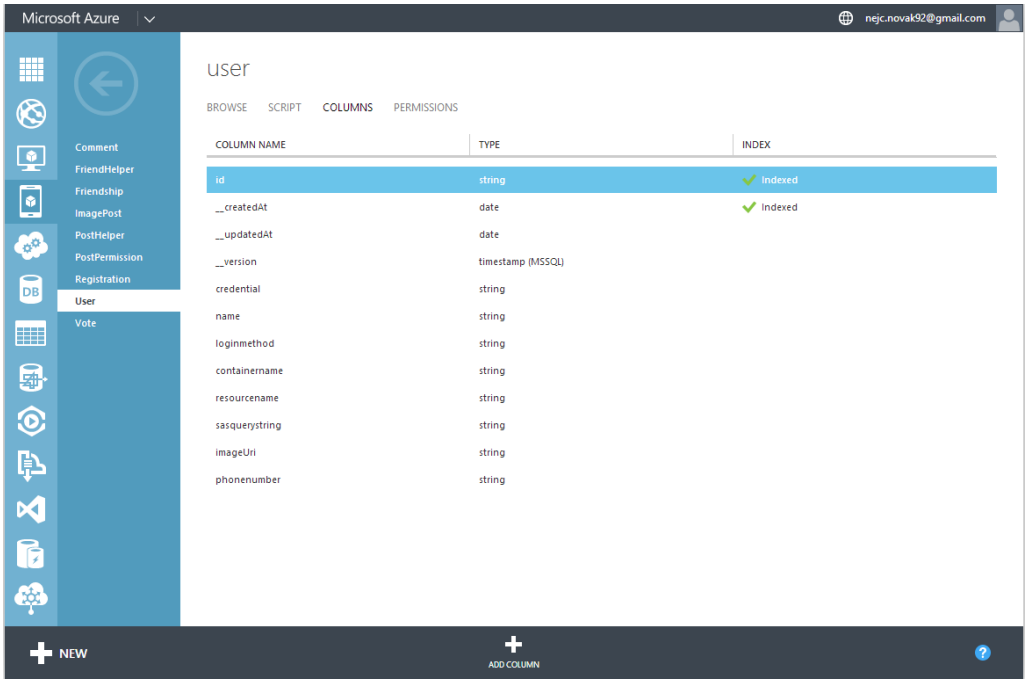
na dejanskih napravah za katere je aplikacija namenjena. Le tako se zanesljivo vidi kako se aplikacija izvaja na ciljnih napravah. Razvojno okolje se lahko zelo dobro prilagodi lastnim potrebam z izpostavitvijo delov, ki so bolj pomembni in pomanjšanjem ali skrivanjem ostalih. Na sliki 2.4 je prikazan urejevalnik kode, datotečna hierarhija aplikacije in grafični oblikovalnik.



Slika 2.5: Orodje Microsoft Blend

2.5 Microsoft Blend

Microsoft Blend je orodje za razvoj in urejanje uporabniških vmesnikov za programe ter spletne in mobilne aplikacije, ki temeljijo na XAML. Visual Studio omogoča predogled in urejanje uporabniških vmesnikov, vendar Blend ponuja več funkcionalnosti in boljšo kontrolo. Zelo olajša ustvarjanje lastnih kontrol in gradnikov na osnovi že obstoječih. Mobilno aplikacijo se lahko hkrati ureja v Visual Studio in Microsoft Blendu, spremembe iz enega orodja pa se samodejno posodobijo v drugem. Na sliki 2.5 je prikazan uporabniški vmesnik orodja Microsoft Blend.



Slika 2.6: Portal za uporavljanje storitev na Microsoft Azure

2.6 Microsoft Azure

Microsoft Azure je platforma za oblačne storitve in aplikacije. Podlaga za razvijanje in objavljanje storitev in aplikacij v oblak so Microsoftova podatkovna središča po celem svetu. Platformo je pod imenom Windows Azure leta 2010 predstavilo podjetje Microsoft. 25. marca 2014 pa so ime Windows Azure spremenili v Microsoft Azure. Platforma podpira množico programskih jezikov, orodij in ogrodij. Podpora širokemu sklopu tehnologij pomeni, da je platforma primerna tudi za razvijalce, ki ne uporabljajo Microsoftovih tehnologij.

Za mobilne aplikacije je namenjen paket Mobilne storitve (ang. Mobile Services). Ta podpira aplikacije na Windows Phone, iOS, Android in tudi HTML5 aplikacije. Vsebuje sklop storitev, ki jih razvijalci potrebujejo v zalednem sistemu aplikacije. Te storitve omogočajo shranjevanje podatkov v bazi, pošiljanje obvestil na mobilne naprave, integracija z družabnimi omrežji, samodejno povečevanje obsega storitev glede na potrebe in integracija z obstoječimi ponudniki identitet [12]. Z njimi lahko upravljamo na portalu za upravljanje storitev (slika 2.6). Ne vsebujejo vseh funkcionalnosti, saj zaledni sistemi aplikacij večinoma niso tako

kompleksni kot zaledni sistemi informacijskih sistemov ali programov za osebne računalnike. Za večino razvijalcev bodo te storitve zadostovale. Razvijanje bo olajšano, saj so najbolj pogoste operacije avtomatizirane in se lahko aktivirajo z nekaj kliki. Hkrati ni zahtevnejših funkcij, ki bi razvijalca lahko zmedle. Razvijalci, ki razvijajo bolj zahtevne aplikacije, pa bodo potrebovali še kakšno dodatno storitev.

Poglavje 3

Razvoj aplikacij

V tem poglavju so opisani deli razvoja in priprave na razvoj aplikacij za mobilni operacijski sistem WP8. Postopki, ki sestavljajo pripravo na razvoj so ključni za uspešne in uporabne aplikacije. Prav tako pa je ključnega pomena oblikovanje izgleda ter strukture. Za dobro uporabniško izkušnjo je priporočljivo upoštevati smernice za izgled aplikacij na WP8. Intuitiven in na pogled privlačen uporabniški vmesnik je prav tako pomemben kot kvalitetna in optimizirana koda aplikacije. Mnogokrat je potrebna tudi uporaba zmogljivosti mobilnih naprav ter povezovanje z zalednim sistemom.

Teoretična razlaga priprave in razvoja aplikacij je obogatena s prikazom na aplikaciji imenovani "Shop or Drop". Po razlagi določenega dela razvijanja je prikazano kako je bil ta del na aplikaciji praktično realiziran. Dodatna razlaga na primeru delujoče aplikacije služi za poglobitev razumevanja in razloženo povezuje v smiselno celoto.

3.1 Priprava na razvoj

Prepogosto se razvijanje oz. kodiranje začne brez ustreznih analiz in postopkov, ki so potrebni pred začetkom izvedbe. Aplikacije so namenjene uporabnikom, zato je treba že pred razvojem upoštevati njihove želje in mnenja. Brez ustrezne raziskave trga in potreb potencialnih uporabnikov bo aplikacija manj primerna. Možno je celo, da je uporabniki sploh ne bodo potrebovali. Zato je priporočljivo, da se te postopke temeljito preuči in kvalitetno izvede. Bolj podrobne informacije so na

voljo tudi v literaturi [2, 3]. Cilj razvijalcev je ustvarjati kvalitetne, popularne in uporabne aplikacije. Za delo pa morajo biti na nek način poplačani, zato sta poleg opisanih postopkov pomembna tudi kvaliteten poslovni model in način zaslužka z aplikacijo.

3.1.1 Analiza problema in zasnova ideje

Za razliko od iger so aplikacije večinoma namenjene reševanju določenega problema. Za uspešno in uporabno aplikacijo, je najprej potrebno zelo dobro definirati in predstaviti problem, ki ga bo reševala. Ugotoviti je potrebno ali je ta problem pri potencialnih uporabnikih prisoten ter kako se trenutno z njim spopadajo. Primerni metodi sta osebni intervju in spletna anketa. Pri obeh je potrebno paziti na tehniko izpraševanja, še posebej pri osebnem intervjuju. Vprašanja morajo biti zastavljena tako, da ne namigujejo odgovorov in da ne usmerjajo intervjuvanca. Intervjuvane osebe ne smejo vedeti kaj se od njih pričakuje in kateri odgovor bi izpraševalec raje slišal. Le tako bodo ugotovitve, povzete iz ankete ali intervjuja, relevantne in uporabne.

Cilj je ugotoviti ali problem obstaja in če se z njim spopada dovolj ljudi. Poleg tega pa je pomembno tudi, da za ta problem še ne obstaja učinkovita rešitev. Če se s problemom sooča le peščica ljudi ali pa zanj že obstaja zadovoljiva rešitev, aplikacije ni smiselno razvijati. V nasprotnem primeru pa se lahko začne načrtovanje rešitve problema. Med raziskovanjem problema in obstoječih rešitev, se je verjetno že pojavila kakšna ideja. Te ideje je potrebno temeljito pregledati in ugotoviti katere so izvedljive ter med njimi identificirati najboljšo.

Pri “Shop or Drop” smo raziskovali na kakšen način se ljudje odločajo, ko nakupujejo oblačila. Nekateri se lahko sami odločijo o nakupu, večina pa si želi slišati mnenje drugih ljudi. Povdarek smo dali na osebne intervjuje, čeprav smo uporabili tudi spletno anketo. Intervjuje smo izvajali v nakupovalnem centru. Tam je velika verjetnost, da najdemo ljudi, ki bi aplikacijo uporabljali. Intervjuvali smo približno 100 uporabnikov pametnih telefonov, starih od 15 do 50 let. Nismo jim razložili zakaj so intervjuvani. Dali smo jim splošna vprašanja in pustili, da so podali svoje mnenje in izkušnje, pri čemer smo poskušali čim manj vplivati na njihove odgovore. Primeri vprašanj, ki smo jih zastavili pri osebnih intervjujih so:

- Kako najraje nakupujete?
- Kako največkrat nakupujete?
- Česa pri nakupovanju ne marate?
- Opišite svoje zadnje nakupovanje. Ali ste naleteli na kakšen problem/težave?
- Ali si ob nakupovanju kdaj pomagate z mobilnikom? Kako?

Ugotovili smo, da večina ljudi raje nakupuje s prijatelji. Ti jim med nakupovanjem lahko svetujejo in nakupovanje tako olajšajo. Ko ljudje nakupujejo sami, se težje odločajo. Dostikrat oblačila ne kupijo, dokler ne dobijo mnenja prijatelja. Ugotovili smo tudi, da je ta problem bolj prisoten pri mlajših osebah. Veliko starejših ljudi že ima ustvarjen stil, katerega se pri nakupovanju držijo, to pa jim omogoča lažje sprejemanje odločitev med nakupovanjem. Starost naše ciljne populacije smo zato znižali in se osredotočili na mlajše osebe. Izpostavljen problem intervjuvanci trenutno rešujejo tako, da se v oblačilu slikajo in sliko delijo s prijatelji. To delajo s pomočjo multimedijskih sporočil, uporabo družabnih omrežij ali elektronskih sporočil. Včasih pa sliko kar v živo pokažejo prijatelju. Vsi ti načini imajo pomankljivosti. Elektronska sporočila, družabna omrežja in kazanje v živo vzamejo preveč časa. Odziv na multimedijska sporočila je veliko hitrejši, vendar sporočila niso vedno zbrana skupaj. Prav tako omogočajo samo interakcijo med pošiljateljem in prejemnikom. Različni prejemniki ne vidijo mnenja ostalih. Različne slike pri nobenem načinu niso zbrane skupaj in se izgubijo med ostalimi sporočili.

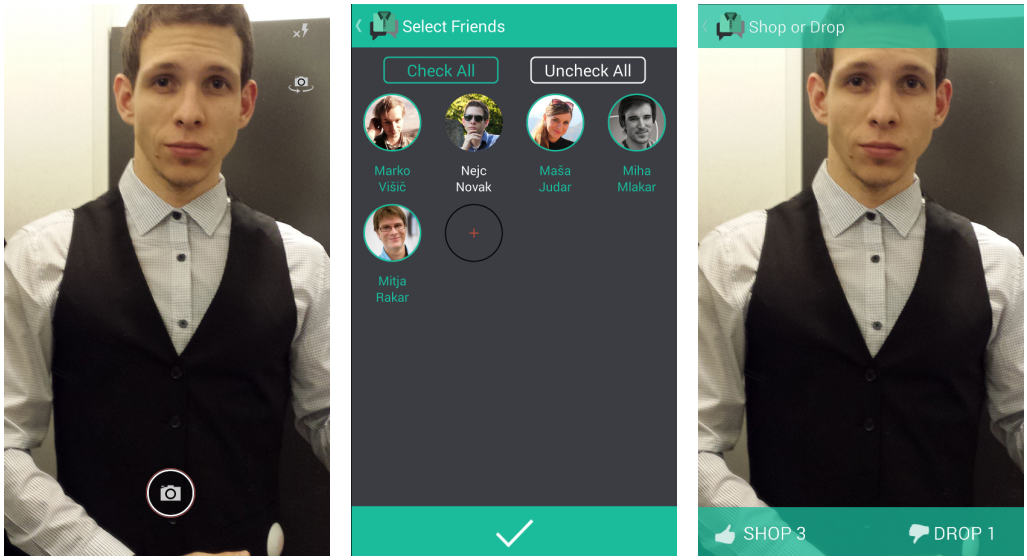
3.1.2 Validacija ideje in prototip

Ideje se pogosto od zasnove do izvedbe temeljito spremenijo. Po identifikaciji najboljše ideje, jo je potrebno validirati. Čim več sprememb na ideji naj se zgodi še pred začetkom razvoja. S tem je lahko prihranjenega veliko časa. Dobro razumevanje ideje in načina realizacije omogoča hitrejši razvoj aplikacije. Spreminjanje že implementiranih funkcionalnosti pa čas razvijanja po nepotrebnem podaljšuje.

Najprej je potrebno preveriti, če obstaja povpraševanje. Za ta namen se lahko uporabi pristajalna stran (ang. landing page). Vsebovati mora dobro predstavitev

naše bodoče aplikacije. Priporočljivo je vključiti tudi okence za vpis elektronskega naslova, za obveščanje o napredku med razvojem in ob izdaji aplikacije. Na tak način se lahko pridobu tudi ljudi, ki bi bili aplikacijo pripravljeni testirati. Nekatera oglaševalska orodja kot sta Google Adwords in Facebook Ads ponujajo kupone za brezplačno testiranje njihovih storitev pred odločitvijo za nakup. S takšnimi kuponi se lahko stran brezplačno oglašuje. Brezplačno se lahko oglašuje tudi na nekaterih forumih, spletnih portalih in podobnih spletnih straneh, ki se ujemajo s tematiko aplikacije. Upoštevati je potrebno, da ponekod ne odobravajo oglaševanja. Priporočljiva je uporaba orodja za spletno analitiko. Na tak način pridobimo podatke o količini ljudi, ki so obiskali stran, kako dolgo so se na strani zadrževali, če so vpisali svoj elektronski naslov ipd. To je ena izmed metod, s katero se lahko v zgodnji fazi preveri ali obstaja interes za aplikacijo. S primernim oglaševanjem in analitiko se lahko doseže in analizira tudi ljudi iz ostalih držav in kontinentov.

Za bolj konkretno preverjanje pa je potrebna priprava prototipa. Za razliko od pristajalne strani, se pri prototipu ne testira samo ideja za rešitev problema. Testira se uporabnost aplikacije, ki je razvita na podlagi te ideje. Osnoven prototip, ki je razvit v začetni fazi je pogosto poimenovan tudi MVP (Minimum Viable Product). Obstaja več vrst prototipov. Ena izmed boljših možnosti je razvoj zelo osnovne različice aplikacije. To je časovno najbolj zahtevno, vendar je najboljši približek dejanske aplikacije. S prototipom se lahko namreč poleg delovanja testira tudi videz in zgradba aplikacije. Za to vrsto prototipa se razvije samo funkcionalnosti, ki so res potrebne, drugače se porabi preveliko časa. Ostale vrste prototipov so še papirnati prototipi (ang. paper prototype), žični okvirji (ang. wireframes) in interaktivni prototipi (ang. clickable prototype). Prototip se vedno testira pri potencialnih uporabnikih. To so tisti ljudje, ki se soočajo s problemom, ki ga rešujemo. Uporabljati morajo platformo oziroma sistem za katerega rešitev razvijamo (v tem primeru WP8). Med uporabo prototipa jih je potrebno opazovati in zapisovati podatke o tem kako se znajdejo in čemu dajejo največji poudarek. Kasneje pa naj povedo kaj ni razumljivo ter kaj bi dodali, odstranili ali spremenili.



Slika 3.1: Prototip “Shop or Drop”

Za “Shop or Drop” smo razvili različne prototipe, vendar smo glavno testiranje opravili z zelo osnovno različico aplikacije. Vključili smo samo najbolj osnovne in pomembe elemente. Določili smo, da so bistvo aplikacije trije koraki s katerimi uporabniki delijo sliko s prijatelji in pridobijo mnenja. Na podlagi dobljenih mnenj se odločijo ali bodo nakup opravili. Prvi korak je slikanje v izbranem oblačilu. V drugem koraku se sliki lahko doda napis in izberejo se osebe, ki bodo prejele sliko. Napis pojasni za kateri kos oblačila na sliki je potrebno mnenje ali doda kontekst sliki. Ta del ni vedno potreben, zato je v prototipu izpuščen. Tretji korak je pošiljanje slike. Zatem lahko uporabnik nadaljuje z nakupovanjem in preizkušanjem oblačil, medtem pa dobi mnenja prej izbranih prijateljev (slika 3.1).

Pri prototipu je uporabnik namesto svojih prijateljev lahko izbiral člane ekipe “Shop or Drop”. Mnenja pa so bila avtomatizirana. Uporabnik je dobil toliko mnenj, kolikor članov je izbral. Mnenja so bila generirana z verjetnostjo 0.75 za pozitivno mnenje in 0.25 za negativno. Mnenja so se pojavila v približno 10 sekundah, med vsakim mnenjem pa je bil naključno določen časovni razmik parih sekund. Tako smo prihranili dosti časa za razvoj prototipa, uporabnik pa je vseeno dobil vtis o končnem delovanju aplikacije. Prototip je bil zelo prepričljiv, saj so nekateri ljudje verjeli, da so člani ekipe “Shop or Drop” res podali mnenja za poslano sliko.

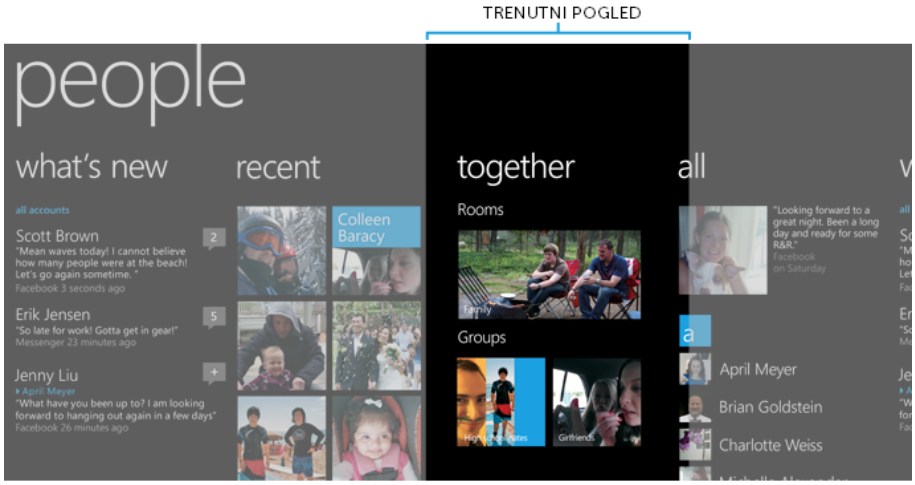
3.2 Dizajn in struktura aplikacije

V sklopu priprave je bilo opravljenega veliko dela. Narejena je bila temeljita analiza problema, identificirane so bile primerne rešitve in način implementacije. Vzpostavljena je bila pristajalna stran za oceno povpraševanja in zbiranje elektronskih naslovov bodočih uporabnikov. Razvit je bil prototip za testiranje in izboljšavo delovanja, strukture in videza aplikacije. V te procese so bili večkrat vključeni tudi potencialni uporabniki. To je dobra podlaga za prvi del razvoja aplikacije, oblikovanje uporabniškega vmesnika. Prototipi so namenoma precej osnovni. Prvi razlog za to je prihranek časa. Drugi razlog pa je, da zaradi tega potencialni uporabniki podajo veliko nasvetov za izboljšave. Nekateri bodo zagotovo povezani z izgledom uporabniškega vmesnika ter z navadami in pričakovanji uporabnikov. Vsi nasveti ne bodo smiselni, izvedljivi in vredni časa, zato je potrebno narediti selekcijo. Aplikacije ne potrebujejo mnogo različnih funkcionalnosti, pomembno je, da so specializirane za reševaje določenega problema. Ugotoviti je treba kateri nasveti dodajajo nepotrebne funkcionalnosti ter kateri dejansko pripomorejo k reševanju problema. Slednje se upošteva pri oblikovanju uporabniškega vmesnika.

V nadaljevanju so opisani gradniki, ki sestavljajo uporabniški vmesnik in kdaj jih je smiselno uporabiti. Uporabniški vmesnik WP8 ima veliko posebnosti, ki jih je potrebno poznati in upoštevati. Razloženi so načini prilagajanja in nadgrajevanja obstoječih gradnikov. S tem se ustvari stil aplikacije, še vedno pa se ohranja konsistentnost z oblikovalskimi smernicami sistema WP8. Opisane so tudi metode, s katerimi je zagotovljeno, da bo aplikacija prilagojena za različne zaslone.

3.2.1 Tipi strani

V WP8 so na voljo trije osnovni tipi strani. Prvi tip je navadna stran, ki je prisotna tudi v uporabniških vmesnikih iOS in Android. Windows Phone pa ponuja še dva posebna tipa, razvita posebej za mobilne naprave. To sta panoramska stran (ang. Panorama) in pivotna stran (ang. Pivot). Vsebujeta več pogledov med katerimi je možno horizontalno premikanje. Večina aplikacij ne vsebuje samo en tip strani, temveč kombinacijo večih.



Slika 3.2: Panoramska stran

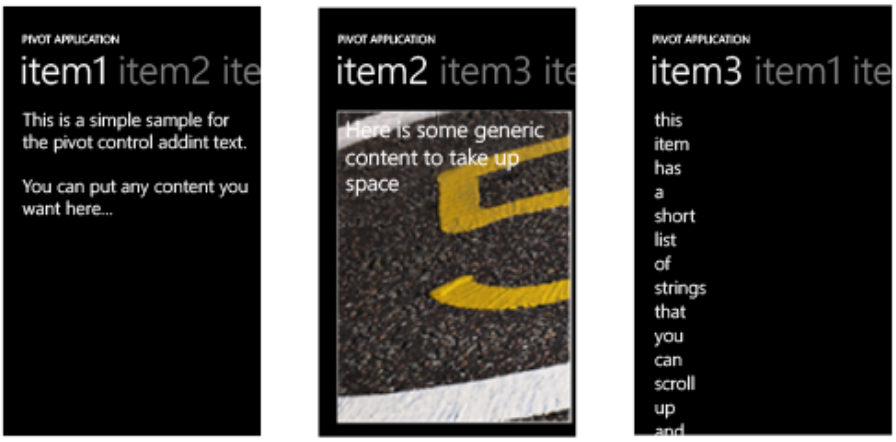
Navadna stran

To je najbolj osnoven tip strani. Dobro se obnese za vpisovanje podatkov, generiranje vsebine, prikaz vsebine, ki potrebuje samo en pogled (slike, posnetki, seznami podatkov in podobno) in za večje količine podatkov. Navadne strani so boljša izbira, če želimo uporabiti gradnike, ki vsebujejo interakcijo s horizontalnimi gestami. Panoramska in pivotna stran namreč takšne geste uporabljata za navigacijo med pogledi.

Panoramska stran

Panoramska stran je sestavljena iz večih pogledov, med katerimi se prehaja horizontalno. Navigacija se izvaja s privzetimi gestami. Na zaslonu je viden celoten pogled in delček desnega pogleda. To uporabniku sporoča, da je navigacija horizontalna in da je na desni strani vsaj še en pogled. Hkrati pa tudi daje vtis neprekinjenosti pogledov panorame. Oblikovalske smernice narekujejo, naj na panoramski strani ne bo več kot 5 pogledov, saj potem stran postane nepregledna. Vsi razdelki uporabljajo panoramsko stran (na sliki 3.2 je razdelek s kontakti).

Panoramske strani so namenjene manjšim količinam podatkov. Pogosto se uporabljajo za predogled osnovnih podatkov, če pa želimo podrobnosti pa nas povezave vodijo do navadih strani, ki so bolj primerne za prikaz večje količine podatkov.



Slika 3.3: Pivotna stran

Pivotna stran

Pivotna stran je tako kot panoramska, sestavljena iz večih pogledov, med katerimi se prehaja horizontalno. Za razliko od panoramske strani, pivotna poglede nalaga dinamično in je zato bolj primerna za prikaz večje količine podatkov. Ker se strani nalagajo dinamično, je na zaslonu naenkrat samo en pogled. Tudi pivotne strani privzeto podpirajo geste, vendar je navigacija mogoča tudi s pritiskom na naslov pogleda. Na zaslonu so poleg naslova pivotne strani prikazani tudi naslovi trenutnega pogleda in naslednjih pogledov (slika 3.3). Pivotne strani večinoma vsebujejo sezname podatkov. Primerne so tudi za prikaz istih podatkov v večih seznamih, ki so različno sortirani. Eden od takšnih primerov je prikaz prijateljev, ki so sortirani glede na lokacijo, skupino ali po kakšnem drugem pravilu.

3.2.2 Gradniki uporabniškega vmesnika

Poleg treh omenjenih vrst strani, ponuja WP8 še druge gradnike. Ti so prisotni tudi na ostalih mobilnih napravah, spletnih straneh ipd. Gradniki so prilagojeni za interakcijo z dotikom, zato so večji kot enaki gradniki na spletu [1]. Interakcija poteka z gestami ali vnosom podatkov preko virtualne tipkovnice. Gradnike delimo v štiri skupine - osnovni gradniki, vsebniki, seznami in vsebinski gradniki [13].

Osnovni gradniki

Osnovni gradniki so najpreprostejši (tabela 3.1). Služijo predvsem vnašanju in prikazovanju podatkov. Najpogosteje so ti podatki v tekstovni obliki. Pri gradniku za vnos gesla so vnešeni znaki prikriti z znakom za maskiranje. Privzeto je uporabljen poln krog (•), vendar se lahko spremeni.

osnovni gradnik	**angleško**	**namen**
tekstovni blok	TextBlock	prikazuje eno ali več vrstic teksta
vnos teksta	TextBox	omogoča vnašanje teksta
vnos gesla	PasswordBox	omogoča vnašanje gesel
drsnik	Slider	omogoča izbiro vrednosti z intervala
vrstica napredka	ProgressBar	prikazuje napredek trenutne operacije

Tabela 3.1: Osnovni gradniki

Vsebniki

Vsebniki so gradniki, ki vsebujejo ostale gradnike (tabela 3.2). Njihov namen je grupiranje in urejanje postavitve vsebovanih gradnikov. Vsebujejo lahko tudi druge vsebnike. Takšno večnivojsko grupiranje je pogosto uporabljeno v aplikacijah. Večina vsebnikov izvira iz razreda Panel, ki omogoča pozicioniranje in urejanje vsebovanih gradnikov. Tudi strani so vsebniki.

vsebnik	**angleško**	**namen**
okvir	Border	omogoča dodajanje okvirja in ozadja gradniku
platno	Canvas	vsebuje gradnike glede na podane koordinate
mreža	Grid	vsebuje gradnike razvrščene v vrstice in stolpce
plošča	StackPanel	vsebuje gradnike v horizontalni ali vertikalni vrsti
drsni pogled	ScrollViewer	omogoča drsno površino za prikaz gradnikov

Tabela 3.2: Vsebniki

Seznami

Obstajata dva gradnika, ki omogočata prikazovanje podatkov v obliki seznama. Navadni seznam (ang. ListBox) omogoča prikazovanje in izbiro podatkov. Uporabnik izbere posamezen element v seznamu s pritiskom nanj. Dolgi seznam (ang. LongListSelector) je drugi, naprednejši tip seznama. Omogoča prikazovanje, grupiranje in izbiro podatkov. Namenjen je seznamom z večjo količino podatkov. Uporabnik lahko uporabi tudi hitro navigacijo do določene skupine podatkov v seznamu, brez drsnega pomikanja. Videz predmetov v seznamih prilagajamo z uporabo podatkovne predloge (ang. DataTemplate).

Vsebinski gradniki

Vsebinski gradniki so podobni vsebnikom, saj omogočajo vsebovanje in prikazovanje vsebine oziroma ostalih gradnikov (tabela 3.3). Njihov namen pa se razlikuje od vsebnikov, katerih edini namen je grupiranje in postavitev vsebine. Vsebinski gradniki izhajajo iz razreda ContentControl, ki omogoča prikaz enega gradnika znotraj samega sebe. Gumb lahko vsebuje na primer besedilo, sliko ali kombinacijo obojega.

vsebinski gradnik	**angleško**	**namen**
gumb	Button	sproži dogodek ob kliku
preklopni gumb	ToggleButton	gumb z dvema stanjema
potrditveno polje	CheckBox	omogoča dve ali tri stanja označenosti
polje za izbiro	RadioButton	omogoča izbiro ene možnosti iz seznama
gumb s povezavo	HyperlinkButton	vsebuje spletno povezavo

Tabela 3.3: Vsebinski gradniki

3.2.3 Gradniki po meri

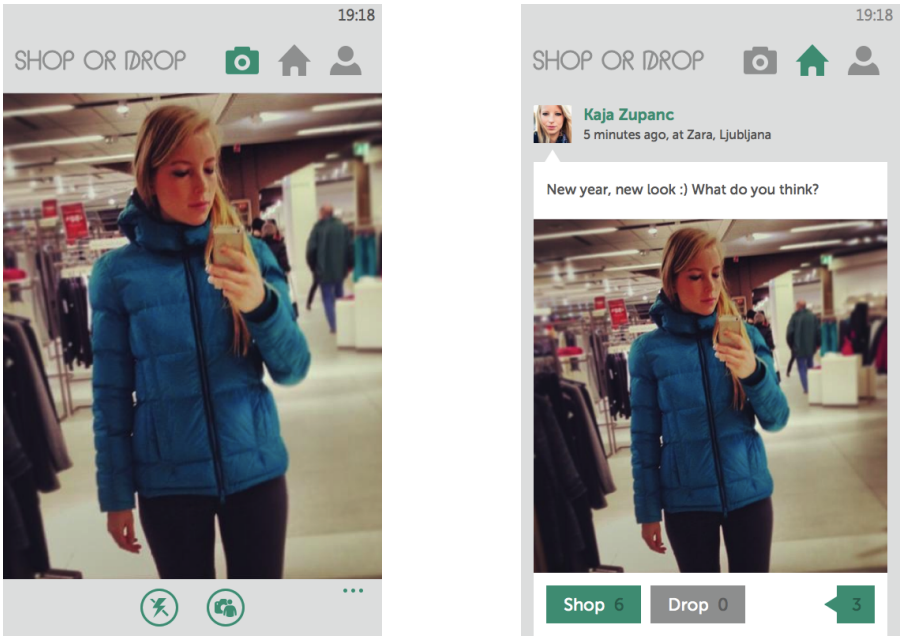
Med oblikovanjem uporabniškega vmesnika je potrebno paziti na dve stvari. Aplikacija mora imeti svoj stil, hkrati pa mora biti njen videz prilagojen za sistem na katerem bo nameščena. Če je ista aplikacija dostopna na večih mobilnih operacijskih sistemih, naj bo stil konsistenten na vseh. Uporabnik mora biti brez izdatnih navodil sposoben uporabljati aplikacijo na sistemu, ki ga je navajen. Za ta namen je dobro uporabiti obstoječe gradnike, ki so jih uporabniki vajeni in jih prilagoditi potrebam aplikacije. Tako se ohrani stil aplikacije in konsistentnost čez več sistemov. Gradniki se lahko prilagodijo samo vizualno ali pa se spremeni tudi njihova funkcionalnost. Vizualne spremembe so manj problematične kot spremembe funkcionalnosti. Uporabniki morajo ob pogledu na gradnik vsaj približno pravilno pričakovati kako se bo obnašal. Horizontalni poteg, ki se običajno uporablja za prehajanje med pogledi, naj ni uporabljen za brisanje vsebine trenutnega pogleda.

Najlažje je gradnike prilagajati v orodju Microsoft Blend. Olajšano je spreminjanje lastnosti gradnikov. Na voljo so obstoječi stili gradnikov, zapisani v XAML kodi, ki jih je mogoče po potrebi spremeniti. Z desnim klikom na element se odpre meni, v katerem se izbere `"Edit template"` in potem `"Edit a copy"`. To omogoča dostopanje do privzetega stila in spreminjanje obstoječega gradnika.

Gradniki po meri v aplikaciji “Shop or Drop”

Pri razvoju aplikacije “Shop or Drop” smo gradnike prilagodili glede na videz in stil aplikacije. Večina gradnikov ima samo spremenjene vizualne lastnosti, kar nekaj pa je tudi gradnikov s spremenjeno funkcionalnostjo.

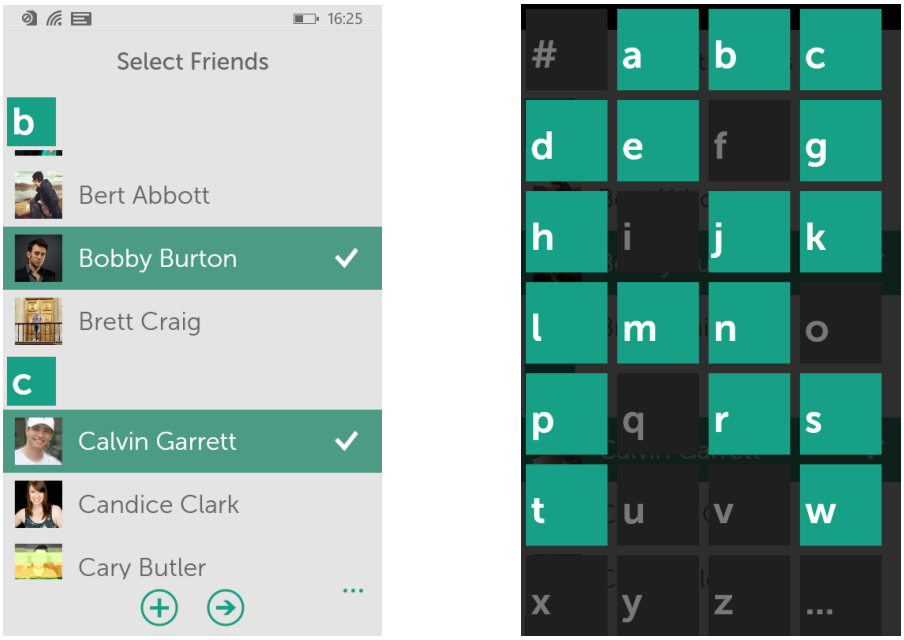
Osrednji gradnik aplikacije je pivotna stran, ki vsebuje tri poglede. V prvem pogledu je na voljo kamera, v drugem so vidne slike, ki so jih ostali delili z nami in v tretjem so slike, ki smo jih mi delili z ostalimi. Za pivotno stran smo se odločili, zato ker je bolj primerna za velike količine podatkov in slik. Za primerjavo lahko uporabimo pivotno stran “Shop or Drop” na sliki 3.4a in primer pivotne strani na sliki 3.3. Po obeh se lahko navigira z gestami. Posebnost naše pivotne strani pa so ikone v desnem zgornjem delu zaslona, ki nadomestijo naslove pogledov. Na tak način se zmanjša glava pivotne strani, kar omogoča boljšo izrabo prostora. Hkrati pa je tak način premikanja med pogledi bolj v stilu s praksami na ostalih sistemih in se lažje vzdržuje konsistentnost aplikacije.



(a) Pivotna stran (b) Gumba po meri

Slika 3.4: Gradniki po meri v “Shop or Drop”

Zanimiva sta tudi gumba za podajanje mnenja Shop ali Drop (slika 3.4b). Od takšnih gumbov so pričakovane tri stvari. Prvič, ko je pritisnjen neoznačen gumb (siva barva), se ta označi (zelena barva). Drugič, ko je pritisnjen označen gumb, se ta odznači. Tretjič, ko je pritisnjen neoznačen gumb, hkrati pa je drug gumb označen, se prej neoznačen gumb označi, drugi gumb pa odznači. Takšnega gradnika v WP8 ni. Obstaja gumb (ang. Button), ki ne ustreza drugi in tretji zahtevi. Potrditveno polje (ang.CheckBox) ne izpolnjuje tretje zahteve. Polje za izbiro (ang. RadioButton) omogoča da je naenkrat izbrana samo ena možnost in tako ustreza tretji zahtevi, vendar ne izpolnjuje druge. Po videzu in funkcionalnosti je najbolj primeren preklopni gumb (ang. ToggleButton), ki pa ne izpolnjuje tretje zahteve. Rešitev v tem primeru je uporaba polja za izbiro z videzom in dodatno funkcionalnostjo preklopnega gumba. To je dosti bolj elegantna rešitev kot uporaba enega izmed obstoječih gradnikov s kodo, ki v ozadju prilagaja delovanje tega gradnika.



(a) Izbiranje prijateljev (b) Hitra navigacija

Slika 3.5: Dolgi seznam za izbiro prijateljev

Uporabili smo tudi prirejen dolgi seznam (ang. LongListSelector) za izbiranje prijateljev. Vsak element v seznamu je vizualno prirejeno potrditveno polje. Seznam omogoča izbiranje večjega števila elementov, ki se ob izbiri obarvajo in označijo s kljukico (slika 3.5a). Vsak element prikazuje slike in imena prijateljev. Ti so grupirani po začetni črki imena. Omogočena je tudi hitra navigacija z izbiro posamezne skupine prijateljev (slika 3.5b).

3.2.4 Žive ploščice

Obstajajo tri različne vrste živih ploščic. Vsaka ima svoj način delovanja. Živa ploščica z obračanjem (ang. Flip Tile) ima sprednji in zadnji del (slika 3.6). Večino časa je prikazan sprednji del, občasno pa se živa ploščica obrne in prikaže se zadnji del s podatki [18]. Ikonska živa ploščica (ang. Iconic Tile) prikazuje logotip in tekst (slika 3.7). Za razliko od ikone z obračanjem, ima ikonska ploščica samo eno stran, vendar je vedno poleg teksta prikazan tudi logotip. Tretji tip je živa ploščica s ciklom. V najmanjši različici vsebuje ta ploščica samo logotip aplikacije [18]. V

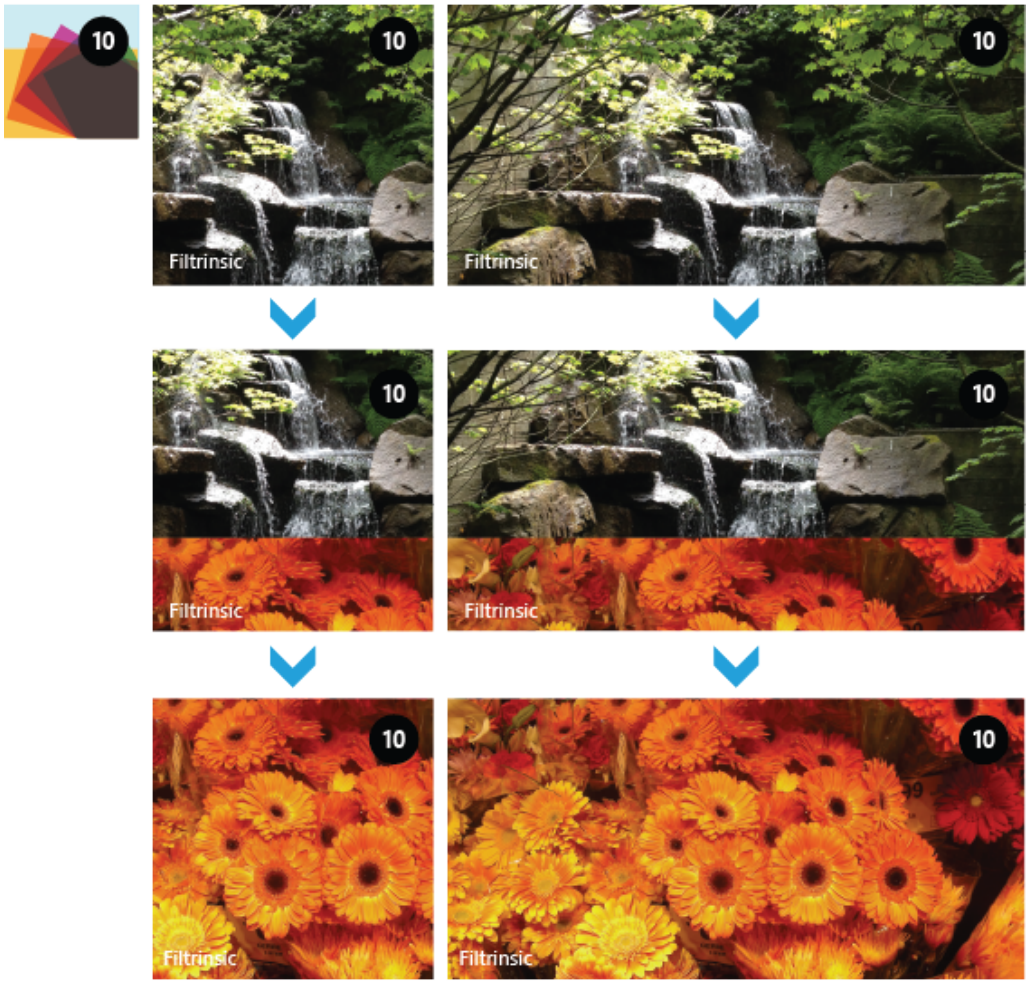
Slika 3.6: Živa ploščica z obračanjem

srednji in večji različici pa prikazuje ime aplikacije in do devet slik, med katerimi kroži kot je prikazano na sliki 3.8.

Vsaka vrsta žive ploščice obstaja v treh različnih velikostih. Velikost posamezne ploščice na telefonu izbira uporabnik. Najmanjša velikost znaša 159 x 159 pikslov, srednja 336 x 336 pikslov ter največja 691 x 336 pikslov. Pri ikonskih živih ploščicah je potrebno paziti, saj mere vsebine niso enake meram ploščice [18].

Pri "Shop or Drop" smo uporabili ikonsko živo ploščico. Spremenili smo barvo ploščice, saj smo želeli, da se ujema z eno izmed glavnih barv aplikacije. Izbrali smo turkizno barvo, ki je v aplikaciji najpogosteje zastopana. Barva ploščice se lahko spremeni z urejanjem datoteke `WMAppManifest.xaml`. Privzeto se v Visual Studio ta datoteka ureja preko uporabniškega vmesnika, ki ne ponuja urejanja barve ploščice. Za spremembo barve ozadja je potrebno direktno spremeniti kodo te datoteke. Če značka `TemplateIconic` že vsebuje značko `BackgroundColor`, ji samo spremenimo vrednost. V nasprotnem primeru pa jo dodamo in vpišemo vrednost, ki je v šestnajstiški obliki ARGB (Alpha Red Green Blue). V našem primeru je `#FF16A085`. `FF` je vrednost transparentnosti, ki je v tem primeru nujna.

Slika 3.7: Ikonska živa ploščica



Slika 3.8: Živa ploščica s ciklom

3.2.5 Prilagajanje različnim zaslonom

WP8 podpira štiri različne resolucije zaslonov. Te resolucije so 480x800 (WVGA), 768x1268 (WXGA), 720x1280 (720p) in 1080x1920 (1080p). Razmerje višine in širine zaslona pri resolucijah WVGA in WXGA je 15:9, pri resolucijah 720p in 1080p pa je razmerje 16:9. Štiri resolucije in dve razmerji zaslona so malenkost v primerjavi s sistemom Android, kjer je ogromno različnih resolucij in razmerij. Vseeno je potrebno paziti, saj lahko uporabniški vmesnik izgleda nepravilno, če je aplikacija razvita brez upoštevanja različnih zaslonov.

Med razvijanjem uporabniškega vmesnika se nikoli ne upravlja z realnimi velikostmi na zaslonu. Uporablja se skalirana resolucija 480x800 pikslov ali 480x853 pikslov za razmerje 16:9. Uporabniški vmesnik in slike, ki bodo prikazane, bodo ob namestitvi aplikacije samodejno prilagojene glede na velikost zaslona mobilnika. Paziti je potrebno, da so slike dovolj velike ločljivosti tudi za največjo resolucijo 1080p. V nekaterih primerih pa je priporočljivo uporabiti slike različnih velikosti. Eden takšnih primerov je slika, prikazana ob nalaganju aplikacije (ang. Splash-Screen). Sistem jo lahko samodejno prilagodi glede na velikost zaslona, vendar včasih izgleda popačeno [14].

Za širino, višino in ostale lastnosti je priporočljivo, da se ne uporabljajo številske vrednosti. Bolje je uporabljati mrežo (ang. Grid) ali ploščo (ang. StackPanel), ki gradnike dinamično prilagajata glede na velikost zaslona. Pri mreži se lahko določi velikost vrstic in stolpcov, pri čemer se uporabljajo vrednosti **Auto** in ***. Vrednost **Auto** samodejno prilagaja velikost vrstice ali stolpca glede na gradnike, ki jih vsebuje. Vrednost * pa označuje kako si bodo vrstice ali stolpci razdelili preostali prostor mreže. Primer uporabe mreže z vrednostmi **Auto** in *:

```
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition Height="*" />
    <RowDefinition Height="2*" />
    <RowDefinition Height="*" />
  </Grid.RowDefinitions>
  ...
  ...
</Grid>
```

V primeru bi prva vrstica zavzela le toliko prostora, kolikor ga potrebujejo gradniki, ki jih vsebuje. Ostale tri vrstice pa bi si razdelile preostali prostor. Količina vrednosti * pove na koliko delov se bo preostali prostor razdelil. Količina vrednosti * pri posamezni vrstici pa koliko delov dobi ta vrstica. Preostali prostor bi se v primeru s slike razdelil na štiri dele. Tretja vrstica bi zavzela polovico preostalega prostora oziroma dva dela, druga in četrta pa vsaka četrtino preostalega prostora oziroma vsaka po en del.

3.3 Uporaba dodatnih zmogljivosti

Ko sta opredeljeni struktura in dizajn aplikacije, je potrebno premisliti še o dodatnih funkcionalnostih, ki bodo vključene v aplikacijo. Z osnovnimi gradniki in funkcionalnostmi je mogoče razviti zelo dobro aplikacijo. Biti mora uporabna in hkrati pregledna. Več funckionalnosti tako ne pomeni vedno, da bo aplikacija boljša. Zato je potrebno ugotoviti kaj uporabniku prinaša prednosti in kaj samo zmanjšuje preglednost. V tem podpoglavju so opisane nekatere najbolj pogosto uporabljene zmogljivosti mobilnih naprav, ki jih je smiselno uporabiti.

3.3.1 Vibriranje

Vibriranje je način s katerim aplikacije omogočajo ustvarjanje povratnih informacij. Pri mobilnih napravah je takšen način še posebej učinkovit. Uporabniki napravo, na kateri je nameščena aplikacija, med uporabo skoraj vedno držijo v rokah. Prekomerna uporaba vibracij je moteča in porablja preveč baterije, zato se z vibriranjem ne sme pretiravati. Smiselno je takrat, ko vizualne povratne informacije niso na voljo ali niso dovolj očitne. Na primer pri manjših gradnikih, kjer s prstom ob pritisku prekrijemo cel gradnik [1]. Za vibriranje sta na voljo razreda `VibrateController` in `VibrateDevice`, ki ponujata določanje časa vibriranja. Ta ne sme presegati 5 sekund [16]. Za testiranje je potrebna fizična naprava, saj emulator ne ponuja testiranja vibriranja. Potrebno pa je tudi upoštevati, da lahko uporabnik vibriranje onemogoči. Pri “Shop or Drop” vibriranja nismo implementirali, saj ne pride do situacij, kjer bi bilo smiselno oziroma potrebno.

3.3.2 Kamera

Na WP8 napravah obstajajo trije načini uporabe kamere. Najbolj osnoven način je uporaba opravila `CameraCaptureTask`. Za bolj zahtevne primere pa razreda `PhotoCamera` in `PhotoCaptureDevice` ponujata večjo prilagodljivost, boljšo uporabniško izkušnjo in več funkcij.

Opravilo CameraCaptureTask

Ko se pokliče to opravilo, se odpre privzeta kamera. Uporabnik naredi sliko, opravilo pa to sliko vrne v aplikacijo. Obstaja tudi možnost, ki omogoči uporabniku, da sliko prej še obreže. Ta način pri "Shop or Drop" uporabljamo za nastavitev ali spremembo uporabniške prikazne slike. Za ta primer uporabe je opravilo primerno, ker je v skladu s smernicami in pričakovanji uporabnika. V naši aplikaciji ta način ni primeren za osrednjo kamero, s katero se uporabnik slika v izbranem oblačilu. Za razliko od spreminjanja prikazne slike, bodo uporabniki zelo pogosto pošiljali slike oblačil. Ta proces mora biti zato čim bolj optimiziran in hiter, uporaba opravila pa bi ga občutno upočasnila.

Razred PhotoCamera

S pomočjo razreda `PhotoCamera` se lahko programsko dostopa do funkcionalnosti kamere. Izbira se lahko med kamerami, v primeru da ima naprava dve kameri (prednjo in zadnjo). Spreminjajo se lahko lastnosti bliskavice, nastavi samodejni fokus ali pa se fokus določi ročno. Izbira se lahko med podprtimi resolucijami in omogoči uporaba fizičnega gumba na napravi za zajemanje slik in videa. Uporaba opravila `CameraCaptureTask` vedno odpre privzeto kamero, kar ne omogoča tako dobre uporabniške izkušnje. Razred `PhotoCamera` pa omogoča umestitev kamere v aplikacijo in do neke mere ponuja prilagajanje zajema slik in videa. Pri "Shop or Drop" nismo uporabili tega razreda, saj ne ponuja dovolj funkcionalnosti za osrednjo kamero, ki je ena pomembnejših elementov aplikacije.

Slika 3.9: Osrednja kamera pri "Shop or Drop"

Razred PhotoCaptureDevice

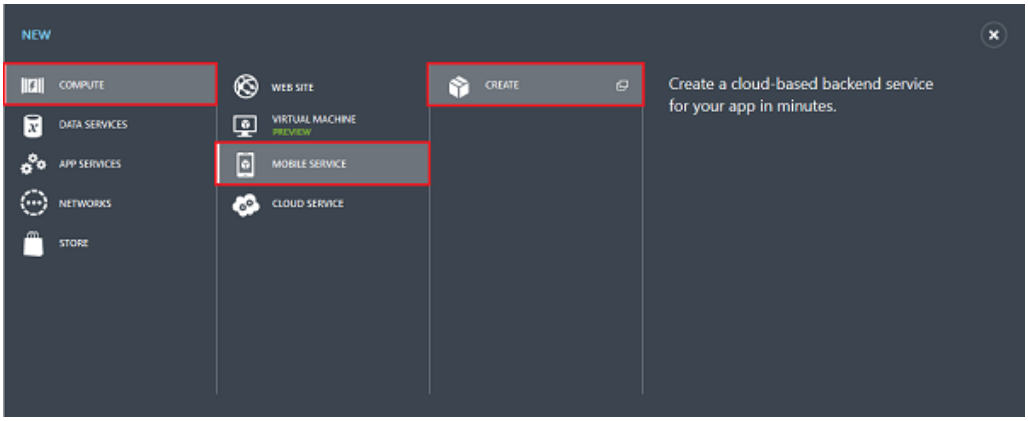
Razred `PhotoCaptureDevice` vsebuje vse funkcije, ki jih ponuja `PhotoCamera`. Prilagajajo pa se lahko tudi bolj napredne lastnosti kamere, kot na primer hitrost zaklopa, vrednost ISO, moč bliskavice, samodejno predvajanje zvoka ob zajemu slik, uravnovešanje beline in določanje parametrov samodejnega fokusa. Pri "Shop or Drop" smo za kamero, s katero uporabnik zajame slike v izbranih oblačilih uporabili razred `PhotoCaptureDevice`. Kamera je v aplikaciji poglavitnega pomena in mora biti dobro konfigurirana. Umeščena je v enega izmed pogledov glavne pivotne strani. Po videzu in uporabi pa je zelo podobna privzeti kameri sistema WP8, saj so takšne kamere uporabniki navajeni in z njo znajo upravljati (slika 3.9).

3.3.3 Merilnik pospeška

Vse novejše mobilne naprave z WP8 imajo večinoma že vgrajen merilnik pospeška (ang. Accelerometer). Ta meri hitrost in smer premikanja naprave v primerjavi z gravitacijo. S pomočjo teh podatkov se lahko ugotovi kako je naprava obrnjena, v katero smer se premika in s kakšno hitrostjo, oziroma kakšna sila deluje na napravo. Te podatke pridobimo s pomočjo razreda `Accelerometer`. Najbolj pomembna lastnost je čas med posodobitvami podatkov. Pri igrah, ki se upravljajo z nagibanjem telefona, bo čas med posodobitvami zelo kratek. Pri ostalih aplikacijah pa je večinoma ta čas lahko daljši. Privzeta vrednost je 2ms. Merilnik pospeška je najbolj smiselno testirati na fizični napravi, vendar je testiranje omogočeno tudi na emulatorju. Če aplikacija podpira različne orientacije, se te menjujejo glede na podatke merilnika pospeška. Pri "Shop or Drop" je smiselna samo pokončna orientacija, tako da merilnika pospeška v aplikaciji ne uporabljamo.

3.4 Povezava z zalednim sistemom

Nekatere aplikacije ne potrebujejo svojega zalednega sistema s podatkovno bazo in ostalimi storitvami. Na primer aplikacije, ki podatke pridobivajo direktno iz spletnih storitev ali pa morajo delovati brez internetne povezave. Nekatere mobilne igre in aplikacije imajo statično vsebino in tako ne potrebujejo povezave z zalednim sistemom. Veliko aplikacij pa vseeno potrebuje vsaj shrambo podatkov v bazi. Če se uporabnik v aplikacijo vpiše z uporabniškim računom, je potrebno uporabniške račune nekje shraniti. Če uporabnik ustvarja vsebino, ki mora biti dostopna iz različnih virov, te vsebine ne moremo preprosto shranjevati na napravi. V tem poglavju so opisane oblačne storitve (Microsoft Azure) in kako se podatki, ki jih shranjujemo v zalednem sistemu primerno prikažejo v aplikaciji. Vseh podatkov ni smiselno shranjevati v podatkovnih bazah, zato je v tem poglavju opisano tudi kako se podatki shranjujejo na mobilnih napravah.



Slika 3.10: Ustvarjanje mobilnih storitev

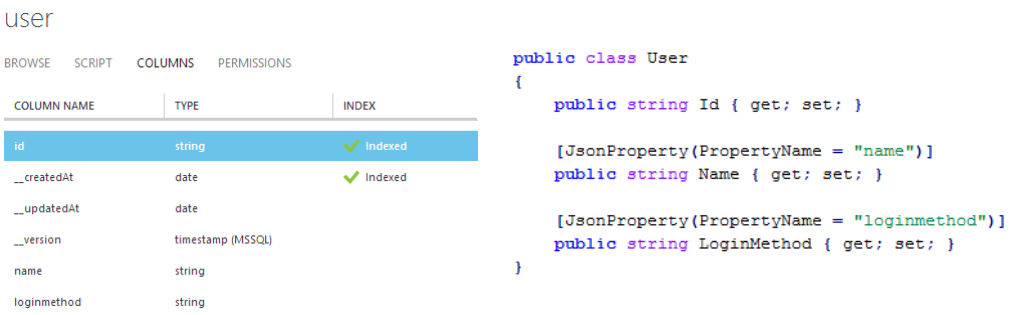
3.4.1 Microsoft Azure

Microsoft Azure ponuja množico oblačnih storitev. Za večino mobilnih aplikacij je najbolj uporaben sklop storitev poimenovan mobilne storitve (ang. Mobile Services). Med drugim ponuja shranjevanje podatkov v podatkovni bazi, pošiljanje obvestil in povezovanje s socialnimi omrežji. Te storitve se samodejno prilagajajo glede na število uporabnikov.

Ta paket smo uporabili tudi pri "Shop or Drop". Poleg vseh omenjenih storitev, smo uporabili še nekatere, ki jih sklop mobilnih storitev ne ponuja. Na primer storitev shrambe BLOB (Binary Large Object), kjer se hranijo vse slike, ki jih uporabniki pošiljajo.

Povezava z mobilnimi storitvami

Mobilne storitve se ustvarijo v portalu za upravljanje s storitvami (slika 3.10). Določiti je potrebno spletni naslov, preko katerega bo aplikacija dostopala do storitev. Naslov je sestavljen iz poljubnega imena in `.azure-mobile.net`. Ustvariti je potrebno tudi novo podatkovno bazo ali izbrati obstoječo. V aplikaciji pa je potreben odjemalec (ang. Client) mobilnih storitev. Z vgrajenim NuGet upravljalnikom paketov se mora namestiti paket `Microsoft.WindowsAzure.MobileServices` v katerem se nahaja razred odjemalca, poimenovan `MobileServiceClient`. Za povezavo odjemalca z mobilnimi storitvami sta potrebna spletni naslov mobilnih storitev in aplikacijski ključ, ki se nahaja v portalu za upravljanje. Preko ustvar-



Slika 3.11: Tabela in pripadajoč razred uporabnika

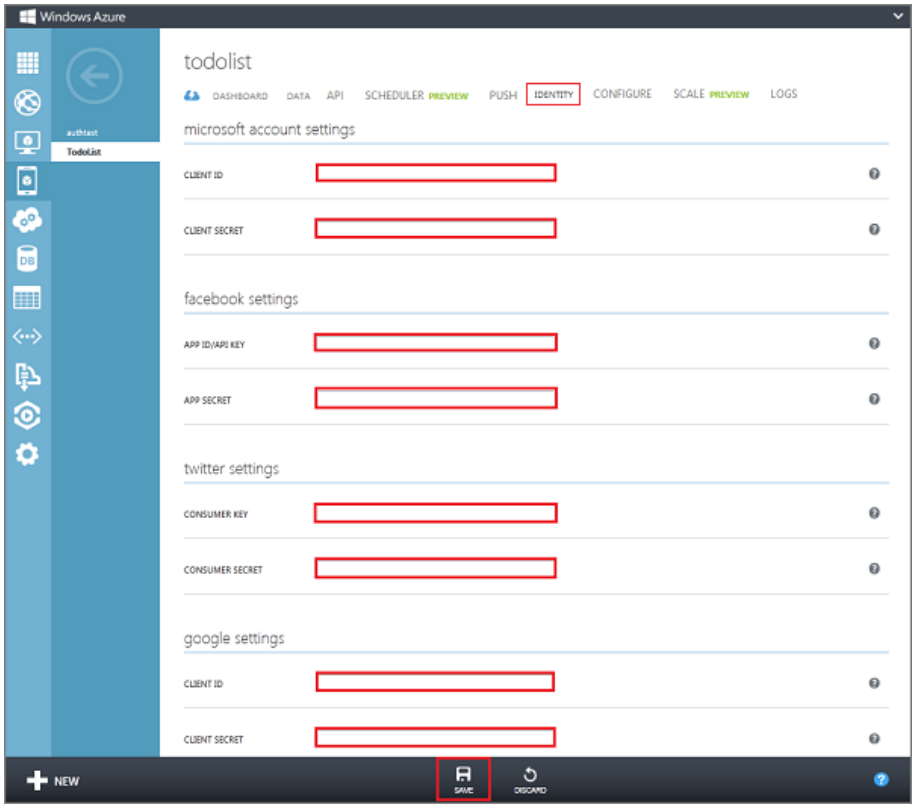
jene povezave je mogoč dostop do podatkovne baze in tabel v njej. Omogočene so osnovne operacije branja, vpisovanja in brisanja podatkov. Vse te operacije so zelo poenostavljene, potrebnih je samo nekaj osnovnih funkcij kot so `GetTable`, `InsertAsync`, `RemoveAsync`, `UpdateAsync` [15].

Tabele je najprej potrebno ustvariti. To se naredi kar v portalu za upravljanje. Uporabniški vmesnik zelo olajša ustvarjanje tabel in dodajanje podatkov. Ko so ustvarjene tabele in struktura njihovih podatkov, se v aplikaciji za vsako tabelo ustvari svoj razred. Na sliki 3.11 je prikazana poenostavljena tabela za vpisovanje podatkov uporabnikov in pripadajoč razred User, ki je ustvarjen v aplikaciji.

Avtentikacija in povezava z družabnimi omrežji

Microsoft Azure ponuja avtentikacijo preko že obstoječih računov. Implementira se lahko avtentikacija z Microsoft, Facebook, Google ali Twitter računom. Pri vsakem ponudniku se dobijo dve vrednosti. Prvo poimenujejo Client, App ali Consumer ID. Druga pa je Client, App ali Consumer Secret. Pridobivanje teh vrednosti je pri vsakem ponudniku malce drugačno. Procesi pridobivanja so opisani v dokumentaciji na spletni strani Microsoft Azure [17]. Pridobljeni vrednosti je potrebno vnesti v portal za upravljanje (slika 3.12). Zatem se v aplikaciji lahko avtenticira uporabnike, kar je prav tako poenostavljeno. Avtentikacijo uporabnika preko Facebook računa se izvede z naslednjim ukazom:

```
MobileServiceUser user = await MobileService.LoginAsync(
    MobileServiceAuthenticationProvider.Facebook);
```



Slika 3.12: Vnašanje pridobljenih vrednosti za avtentikacijo

Poleg avtentikacije, ponuja povezovanje z obstoječimi računi uporabnikov tudi ostale prednosti. Če se uporabnik na primer poveže s svojim Facebook profilom, lahko aplikacija dostopa do njegovih podatkov. V tem primeru so ti podatki Facebook ID, ime, spol, starost, uporabniška slika, elektronski naslov in uporabnikovi prijatelji. Za dostop do bolj podrobnih podatkov pa je potrebno uporabnika posebej prositi za dovoljenje [19].

Če so v aplikaciji vsebine, ki jih lahko vidijo samo avtenticirani uporabniki, se lahko v portalu za upravljanje tabelam nastavijo pravice za dostop do podatkov. Vsaka tabela ima štiri pravice - za branje, pisanje, posodabljanje in brisanje podatkov. Za različne tabele bodo potrebne različne kombinacije dodeljenih pravic. Pri nekaterih aplikacijah imajo tudi neavtenticirani uporabniki dostop do določenih vsebin. Pri “Shop or Drop” se uporabniki lahko vpišejo in avtenticirajo s pomočjo že ustvarjenega Facebook računa. Edina pravica, ki jo imajo neavtenticirani uporabniki je ustvarjanje uporabniškega računa.

3.4.2 Vezava podatkov

Večino podatkov, pridobljene iz zalednega sistema je potrebno na nek način prikazati uporabniku. To je najlažje doseči z vezavo podatkov v gradnike uporabniškega vmesnika. Vežejo se lahko tudi podatki, ki niso pridobljeni iz podatkovne baze. V gradniku za vnos teksta se lahko z XAML kodo prikaže ime uporabnika:

```
<TextBox Text="{Binding Name}" />
```

Ta koda sama po sebi ne bo delovala, saj mora biti gradnik povezan z razredom, ki vsebuje ime uporabnika. To se doseže z nastavitvijo podatkovnega konteksta (ang. DataContext) na razred uporabnika, ki je bil ustvarjen ob povezavi s podatkovno bazo. Podatkovni kontekst se lahko nastavi gradniku ali enemu izmed njegovih staršev. Pri vezavi se bo najprej vedno preveril kontekst samega gradnika. Če ustrezen kontekst ne bo najden, se bo preveril kontekst starša in tako naprej do vrha XAML drevesa. Če je potrebno v večih gradnikih za vnos teksta prikazati različne podatke uporabnika, se lahko te gradnike postavi v vsebnik (na primer v mrežo) in temu vsebniku nastavi podatkovni kontekst. Tako ni potrebno nastavljati konteksta vsakemu gradniku posebej. Tudi če je potrebno kontekst spremeniti, je bolje da je definiran samo na enem mestu.

Na voljo so trije načini vezave podatkov. Najosnovnejši način je `OneTime`, ki vrednost v viru preveri samo ob prvem prikazu gradnika. Če se mora vrednost v gradniku ob spremembah v viru posodabljati, je potrebno uporabiti način `OneWay`. V nekaterih primerih pa je dobro, da se vrednost posodablja tudi v obratno smer. Če se spremeni vrednost v gradniku za vnos teksta, se mora v tem primeru spremeniti tudi vrednost v viru. To nam omogoča način `TwoWay`. Za načina, pri katerih se vrednosti posodabljajo, mora vir podatkov uporabljati vmesnik `INotifyPropertyChanged`, ki poskrbi za obveščanje o spremembah. Tako bi izgledal prejšni primer, razširjen še z načinom vezave:

```
<TextBox Text="{Binding Name, Mode=TwoWay}" />
```

Podatki so lahko vezani na katerikoli atribut gradnika in hkrati tudi na več gradnikov. Razširjen primer bi lahko imel še vezavo za barvo teksta:

```
<TextBox Text="{Binding Name, Mode=TwoWay}"
         Foreground="{Binding FavouriteColor}" />
```

3.4.3 Shranjevanje podatkov na napravi

Vseh podatkov ni smiselno shranjevati v zalednem sistemu. Nekateri podatki so vezani na napravo ali pa morajo biti na voljo ne glede na internetno povezavo. Zato so shranjeni na napravi. Obstajata dva načina shranjevanja podatkov na napravi. Pri izolirani hrambi (ang. Isolated Storage) so podatki shranjeni v obliki parov ključ-vrednost ali pa v obliki map in datotek. Drug način shranjevanja podatkov pa je v lokalni podatkovni bazi. Do shranjenih podatkov se lahko pri obeh načinih dostopa samo iz aplikacije, ki je te podatke shranila. Če uporabnik aplikacijo izbriše z naprave, se bodo hkrati izbrisali tudi vsi podatki.

Izolirana hramba

Izolirana hramba je enostavnejši način shranjevanja podatkov na napravi. Podatki se lahko lokalno shranjujejo v dveh oblikah. Najpreprostejša oblika so pari ključ-vrednost. Za to se uporabi razred `IsolatedStorageSettings`. Shranjujejo se lahko vsi objekti, ki imajo možnost serializacije (ang. Serialization). To je pretvorba objektov v niz bajtov. Uporablja se za shranjevanje stanja objektov. Podatki se ob shranjevaju samodejno serializirajo, ob dostopanju pa se deserializirajo [1]. Ta način je uporaben za shranjevanje nastavitev ali vrednosti v gradnikih ob zapiranju aplikacije. Ob ponovnem odprtju se lahko te vrednosti vstavijo nazaj v gradnike. `IsolatedStorageSettings` predstavlja navaden slovar (ang. Dictionary), tako da se podatki shranjujejo in dostopajo na enak način kot pri navadnem slovarju:

```
IsolatedStorageSettings settings = IsolatedStorageSettings.
   ApplicationSettings;
settings.Add("userPreference", pref.Text);
string pref = settings["userPreference"] as string;
settings.Remove("userPreference");
```

Drug način shranjevanja je v obliki map in datotek. Za to se uporabljata razreda `StorageFolder` in `StorageFile`. S tem načinom se ustvarjajo, dostopajo in brišejo datoteke (ang. File) in mape (ang. Folder). Za pisanje podatkov v datoteke je potrebno več kode kot pri prejšnem načinu. Uporaba je smiselna, če se shranjujejo večje količine podatkov.

Podatke iz prejšnega primera, bi se v datoteko shranilo na tak način:

```
byte[] fileBytes = UTF8.GetBytes(pref.Text.ToCharArray());
StorageFolder local = ApplicationData.Current.LocalFolder;
StorageFolder folder = await local.CreateFolderAsync("pr",
    CreationCollisionOption.OpenIfExists);
StorageFile file = await folder.CreateFileAsync("pr.txt",
    CreationCollisionOption.ReplaceExisting);
using (Stream s = await file.OpenStreamForWriteAsync())
{
    s.Write(fileBytes, 0, fileBytes.Length);
}
```

Pri "Shop or Drop" smo uporablili izolirano hrambo za uporabniške nastavitve ter za shranjevanje stanja aplikacije. Slike, ki jih uporabniki pošiljajo in prejemajo, pa se v aplikaciji shranjujejo samo začasno.

Lokalna podatkovna baza

Poleg izolirane hrambe je voljo tudi shranjevanje podatkov v lokalni podatkovni bazi. Najprej je potrebno premisliti ali je takšna baza res potrebna. Podatkovne baze namreč zasedejo več prostora kot shranjevanje podatkov v datoteke. Dva najpogostejša razloga za shranjevanje podatkov pri aplikacijah sta shranjevanje podatkov ob zaprtju aplikacije, za namen ponovnega prikaza ob odprtju ter lokalno shranjevanje nastavitev aplikacije. Za ta dva namena ni potrebe po lokalni podatkovni bazi, ki podpira povpraševanje in posodabljanje podatkov. Pogosto pa se uporabljajo pri aplikacijah ali igrah, ki nimajo dostopa do zalednega sistema.

Lokalna podatkovna baza je kompaktna različica baze SQL (Structured Query Language). Kljub temu za povpraševanje ne uporabljamo jezika SQL. Uporabljamo poizvedbe LINQ (Language-Integrated Query), ki se v ozadju prevedejo v parametrizirane poizvedbe SQL. Poizvedba LINQ, ki poišče vse uporabnike, kateri imajo manj kot 10 objav in jih sortira po imenu bi izgledala takole:

```
var query = from u in users
            where u.Posts <= 10
            order by u.Name
            select u;
var results = query.ToList();
```

Za uporabo lokalne podatkovne baze je, podobno kot pri podatkovni bazi v zalednem sistemu, potrebno ustvariti razrede, ki predstavljajo tabele v bazi. Potrebno je ustvariti tudi kontekstni razred, ki izhaja iz razreda `DataContext`. Ta razred predstavlja dostopno točko podatkovne baze in upravlja vse spremembe. Razredi, ki predstavljajo tabele, so dostopni preko kontekstnega razreda. Pri "Shop or Drop" nismo uporabljali lokalnih podatkovnih baz, saj je za nas bolj primerno shranjevanje v izolirani hrambi. Več informacij o lokalnih podatkovnih bazah, LINQ, indeksiranju in optimizaciji je na voljo v literaturi [1, 20].

Poglavje 4

Nadaljnji koraki

V tem poglavju so opisani procesi, ki pridejo na vrsto po koncu razvijanja. Pred izdajo aplikacije, jo je potrebno testirati in odpraviti hrošče. Po oddaji na trg aplikacij Windows Phone Store, pa mora biti odobrena za izdajo. Opisane so najbolj pogoste napake, zaradi katerih aplikacije niso sprejete na trg. Samo izdaja pa večinoma ni dovolj. Da izdana aplikacija doseže čim več ljudi iz ciljne populacije, jo je potrebno tržiti. Eden bolj pomembnih korakov pa je tudi analitika, ki mogoča pregled nad uporabo aplikacije in izpostavi šibke točke aplikacije ter trženja.

4.1 Testiranje

Kljub testiranju prototipa in testiranju delovanja funkcionalnosti med samim razvijanjem, se na koncu večinoma vseeno pojavi kakšen hrošč. Aplikacijo je potrebno pred izdajo temeljito testirati. Poleg članov razvojne ekipe naj aplikacijo testira čimveč ljudi iz ciljne publike. To so potencialni uporabniki, ki bodo aplikacijo kasneje dejansko uporabljali. Vsako odkrito napako naj podrobno opišejo. Opis naj vključuje vsaj specifikacije uporabljene naprave in sistema, okoliščine v katerih je prišlo do napake in kaj so hoteli doseči medtem ko so naleteli na napako. Opišejo naj tudi nepričakovano delovanje in stvari, ki jih niso razumeli. Pomembno je, da lahko uporabnik že samo ob pogledu na uporabniški vmesnik oceni kako bo njegova interakcija vplivala na aplikacijo. Uporabi navodil se v večini primerov da izogniti. Navodila so smiselna samo ob funkcijah ali gradnikih, ki uporanikom ne bodo znani. Število takšnih funkcij in gradnikov naj bo čim manjše.

4.2 Izdaja aplikacije

Ko so odstranjeni vsi hrošči, se aplikacija lahko izda na Windows Phone Store. Preden se aplikacija odda na preverjanje primernosti za izdajo, je potrebno prebrati in upoštevati zahteve in priporočila [21]. Če aplikacija ne ustreza vsem zahtevam, je lahko izdaja zavrnjena. Najpogostejše napake, zaradi katerih je izdaja zavrnjena, so:

- v opisu so jeziki, ki v aplikaciji niso podprti
- v aplikaciji ni možnosti za izklop lokacijskih storitev
- aplikacija z izklopljenimi lokacijskimi storitvami ni uporabna
- oddane zaslonske slike ne vsebujejo samo grafike aplikacije, temveč tudi razhroščevalno vrstico, okvir telefona ipd.
- oddane so samo zaslonske slike za en jezik, ne za vse podprte
- uporabljene so privzete ikone aplikacije
- aplikacija se nepričakovano in brez opozorila zapre
- aplikacija potrebuje več kot 5 sekund, da prikaže prvi zaslon
- aplikacija potrebuje več kot 20 sekund, da postane odzivna
- manjkajo podatki, ki so potrebni za prijavo v aplikacijo
- nepravilno delovanje gumba za navigacijo nazaj
- aplikacija ne podpira svetle in temne teme
- manjkajo tehnične informacije kot so ime in verzija aplikacije, kontakt za podporo, informacije o razrešenih hroščih, ipd.

4.3 Analitika

Za razumevanje uspeha ali neuspeha aplikacije je potrebna analitika. Obstaja veliko orodij, ki ponujajo analitiko za mobilne aplikacije. Z njihovo pomočjo se zbirajo podatki o uporabi in delovanju aplikacije. Uporabijo se lahko za sprejemanje boljših odločitev, ki so podprte z dejstvi, ne ugibanjem. S podatki se lahko tudi meri uspešnost aplikacije oz. doseganje zadanih ciljev. Da je uporaba analitike uspešna, je potrebno definirati, kaj za izbrano aplikacijo pomeni uspeh in izbrati cilje. Določeno število prenosov v prvem tednu ali mesecu? Razmerje med številom prenosov in nakupov v aplikaciji? Povprečen zaslužek na uporabnika? Ocena aplikacije? Vse to so ključni pokazatelji uspešnosti (ang. Key Performance Indicator oz. KPI). Glede na izbrane pokazatelje se aplikacija nadgrajuje in izboljšuje. Pri tem je potrebno opazovati kako posamezna nadgradnja vpliva na pokazatelje. Spodaj so našteti primeri nekaterih pogosto uporabljenih ključnih pokazateljev uspešnosti [22, 23]:

Število prenosov in posodobitev

Koliko ljudi je preneslo aplikacijo, koliko jih je aplikacijo zbrisalo iz svoje naprave in koliko jih uporablja najnovejšo različico v primerjavi s tistimi, ki uporabjajo starejše, neposodobljene različice.

Število odpiranj aplikacije

Kolikokrat so uporabniki aplikacijo odprli in koliko uporabnikov je aplikacijo odprlo. Nekateri uporabniki bodo aplikacijo prenesli, vendar je ne bodo nikoli ali redko odprli. Na tak način se ugotovi število aktivnih uporabnikov.

Trajanje uporabe aplikacije

Kako dolgo traja posamezna seja uporabe aplikacije. To je eden pomembnejših pokazateljev, saj prikazuje angažiranost uporabnikov (ang. User Engagement).

Povprečen čas med sejami

Koliko časa mine do naslednje uporabe aplikacije (ang. Recency). Tako se ugotovi kako pogosto uporabniki uporabljajo aplikacijo.

Povprečna ocena in število mnenj

Windows Phone Store ponuja, tako kot ostali trgi za aplikacije, ocenjevanje aplikacij in podajanje mnenj. Povprečna ocena pove kako popularna je aplikacija med uporabniki.

Razmerje med novimi in starimi uporabniki

Koliko novih uporabnikov je preneslo aplikacijo (ang. Acquisition) glede na stare uporabnike v določenem času (na primer v enem mesecu). Tako se ugotovi hitrost rasti aplikacije (ang. Growth Rate).

Zadrževanje uporabnikov

Koliko obstoječih uporabnikov aplikacija zadrži (ang. User Retention) v določenem časovnem obdobju, oziroma koliko uporabnikov v tem obdobju izgubi (ang. Churn Rate).

Konverzija

Če je v aplikaciji določen cilj (prijava, oddaja ocene, nakup) je pomembno vedeti, pri kolikih uporabnikih je ta cilj uresničen (ang. Conversion).

Povprečen zaslužek na uporabnika

Kolikšen je zaslužek na povprečnega uporabnika, ki aplikacijo uporablja vsaj določen čas.

Čas zagona in nalaganja

Koliko časa aplikacija potrebuje, da se zažene in v celoti naloži. Če je ta čas prevelik, bodo uporabniki obupali in aplikacijo zaprli.

Način uporabe

Kakšen je potek uporabe aplikacije in navigacije po aplikaciji (ang. Behaviour Flow). Katere funkcije se največ uporabljajo in katere najmanj (ang. Event Flow).

Demografika uporabnikov

Od kod prihajajo uporabniki in kakšne naprave uporabljajo. Kje je največ uporabnikov in kje so uporabniki najbolj aktivni. Upoštevati je potrebno navade in kulturo uporabnikov.

4.4 Trženje aplikacije

Trženje aplikacije je lahko zelo učinkovito, če je podprto z analitiko. Obstajajo različni načini zaslužka z aplikacijo. Najpogostejši so plačljiva aplikacija, nakupi znotraj aplikacije (ang. In-App Purchase) in oglaševanje znotraj aplikacije. Pri vseh teh načinih je pomembno, da ima aplikacija čim več uporabnikov, to pa se doseže z dobrim trženjem. Opazovati je potrebno kateri načini trženja so efektivni in s katerim se najbolje doseže čim več potencialnih uporabnikov. Aplikacije ki so na vrhu svoje kategorije ali pa so izbrane in posebej predstavljene (ang. Featured), s takšno izpostavljenostjo pridobijo ogromno novih uporabnikov. Takšnih aplikacij je relativno malo. Vseeno pa je pomembno, da lahko uporabniki na trgu aplikacijo hitro najdejo. Za povečanje števila uporabnikov, ki aplikacijo najdejo na Windows Phone Store, je potrebno optimizirati opis in predstavitev aplikacije. Takšna optimizacija se imenuje ASO (Application Store Optimization). Število prenosov se povečuje s postopnimi spremembami in opazovanjem analitike. Na tak način se lahko ugotovi kako na število prenosov vplivajo sprememba imena, logotipa, opisa ipd. Te stvari se lahko spreminjajo ob posodobitvi aplikacije na trgu.

Veliko uporabnikov aplikacije pridobijo iz zunanjih virov, torej jih uporabniki ne najdejo direktno na trgu mobilnih aplikacij. Najočitnejši način za pridobivanje uporabnikov je oglaševanje na spletu, plakatih, panojih ipd. Obstajajo pa tudi ostale, cenejše rešitve. Postavitev spletne strani in uporaba družabnih omrežij sta dva zelo pomembna načina, saj uporabnikom omogočata, da aplikacijo lažje najdejo. Uporaba družabnih omrežij služi tudi za izgradjo skupnosti, interakcijo z uporabniki in obveščanje o spremembah, novih aplikacijah ipd. Nekateri izmed ostalih načinov so še oglaševanje preko ostalih aplikacij iz istega ali podobnega področja ter spodbujanje uporabnikov k podajanju mnenja in ocene. Če je aplikacija plačljiva, je lahko učinkovito začasno znižanje cene, saj nekatere spletne strani spremljajo padce v cenah aplikacij in jih izpostavljajo potencialnim uporabnikom. Izpostavljenost pa se lahko poveča s pridobitvijo ocen ali člankov na spletnih straneh z velikim številom ogledov ali omembe na popularnih profilih družabnih omrežij.

Poglavje 5

Sklepne ugotovitve

Diplomsko delo predstavlja celoten proces izdelave aplikacij za mobilni operacijski sistem WP8. Povdarek je na posebnostih, ki jih razvijalci za ostale operacijske sisteme niso vajeni. Vključeni so tudi procesi, katerim razvijalci pogosto ne posvetijo dovolj časa, vendar so zelo pomembni za izdelavo uspešnih aplikacij. Ti procesi so analiza problema, zasnova in validacija ideje, raziskava trga, izdelava prototipa, trženje aplikacije, ustrezna uporaba analitike in komunikacija z uporabniki skozi celoten proces izdelave aplikacije. Podatki o teh procesih in razvoju za WP8 so bili zbrani iz različnih virov. Izbrani in predstavljeni so najbolj pomembni deli, podprti pa so s primeri iz aplikacije "Shop or Drop", ki je bila razvita po opisanih smernicah.

Družina operacijskih sistemov Windows Phone vsebuje veliko posebnosti in prednosti, ki so posledica relativno majhne starosti. Razvijalci za ostale operacijske sisteme na te posebnosti niso navajeni, vendar se jih lahko hitro privadijo. V diplomskem delu so zbrane, ker je njihovo poznavanje ključno za razvoj aplikacij. Uporabnikom je potrebno zagotoviti dobro izkušnjo. Zato je potrebno pri oblikovanju in programiranju aplikacije upoštevati posebnosti in smernice, da bo aplikacija uporabniku razumljiva in bo naletel na manj problemov.

Izdelava aplikacij je kompleksen postopek. Diplomsko delo predstavlja celoten postopek izdelave, ki občutno poveča možnosti za uspeh aplikacije. Kljub dobri pripravi in izvedbi pa ne bo vsaka aplikacija uspela. Pomembno je, da je postopek izdelave kvaliteten, saj se je možno tudi iz neuspeha marsikaj naučiti. S primerno komunikacijo z uporabniki se lahko ugotovi ali je neuspeh posledica aplikacije. Če

pa je neuspeh posledica slabega trženja, pa bo to razvidno iz analitike. Pomembno je identificirati razlog neuspeha, da se lahko ob izdelavi naslednjih aplikacij temu izogne.

Predstavljene teme so obsežne. V diplomskem delu so opisane samo tiste, ki so najbolj pogosto uporabne. Dosti pa je še podrobnosti in tem, ki so uporabne v bolj specifičnih primerih. Smernice in predlogi za nadaljno delo so zato raziskovanje in poglabljanje znanja o opisanih temah. Pomembno pa je, da se naučeno preverja tudi v praksi, saj se lahko iz lastnih izkušenj in napak največ naučimo.

Literatura

[1] Shawn Wildermuth, *Essential Windows Phone 8*, Addison-Wesley, 2013, pogl. 1, 4, 7, 8, 9.

[2] Eric Ries, *The Lean Startup*, Crown Business, 2011, pogl. 3.

[3] Ash Maurya, *Running Lean*, O'Reilly Media, 2012, pogl. 3.

[4] Windows Phone Preview for Developers. Dostopno na: `http://dev.windowsphone.com/en-us/develop/devpreview`

[5] Gartner: Annual Smartphone Sales. Dostopno na: `http://www.gartner.com/newsroom/id/2665715`

[6] Microsoft by the Numbers. Dostopno na: `http://www.microsoft.com/en-us/news/bythenumbers/index.html`

[7] System requirements for the emulator for Windows Phone 8. Dostopno na: `http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff626524(v=vs.105).aspx`

[8] C# Language Specification, 4th Edition, 2006. Dostopno na: `http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf`

[9] TIOBE Index for July 2014. Dostopno na: `http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html`

[10] XAML Overview (WPF). Dostopno na: `http://msdn.microsoft.com/en-us/library/ms752059(v=vs.110).aspx`

[11] Visual Studio Features. Dostopno na: `http://www.visualstudio.com/explore/features-overview-vs`

[12] Azure Features. Dostopno na: https://azure.microsoft.com/en-us/solutions/

[13] Controls for Windows Phone 8. Dostopno na: http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff402561(v=vs.105).aspx

[14] Multi-resolution apps for WP8. Dostopno na: http://msdn.microsoft.com/en-us/library/windowsphone/develop/jj206974(v=vs.105).aspx

[15] Get started with data in Mobile Services. Dostopno na: http://azure.microsoft.com/en-us/documentation/articles/mobile-services-windows-phone-get-started-data/

[16] How to vibrate the phone for WP8. Dostopno na: http://msdn.microsoft.com/en-us/library/windows/apps/jj206994(v=vs.105).aspx

[17] Get started with authentication in Mobile Services. Dostopno na: http://azure.microsoft.com/en-us/documentation/articles/mobile-services-windows-phone-get-started-users/

[18] Tiles for Windows Phone 8. Dostopno na: http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh202948(v=vs.105).aspx

[19] Permissions with Facebook Login. Dostopno na: https://developers.facebook.com/docs/facebook-login/permissions/v2.0

[20] Local database for WP8. Dostopno na: http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh202860(v=vs.105).aspx

[21] App certification requirements for Windows Phone. Dostopno na: http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh184843(v=vs.105).aspx

[22] 9 Mobile App KPIs to Know. Dostopno na: http://mashable.com/2013/09/04/mobile-app-metrics/

[23] 7 KPIs of Mobile Apps. Dostopno na: http://www.atomrain.com/it/technology/7-key-performance-indicators-mobile-apps